Angular aplikacija stranica za upravljanje građevinskom tvrtkom

Ćosić, Marko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

ANGULAR APLIKACIJA STRANICA ZA UPRAVLJANJE GRAĐEVINSKOM TVRTKOM

Završni rad

Marko Ćosić

Osijek, 2022.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 12.09.2022.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Marko Ćosić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R4188, 23.07.2018.
OIB Pristupnika:	94691007121
Mentor:	Izv. prof. dr. sc. Ivica Lukić
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Angular aplikacija stranica za upravljanje građevinskom tvrtkom
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak završnog rad:	Aplikacija treba omogućiti upravljanje građevinskom tvrtkom. Od vođenja djelatnika, opreme, aktivnih projekata i slično. Za svaki projekt treba voditi djelatnike na projektu, potrebne resurse i opremu. Rezervirano za: Marko Ćosić
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	12.09.2022.
Datum potvrde ocjene od strane Odbora:	

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 15.09.2022.

Ime i prezime studenta:	Marko Ćosić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4188, 23.07.2018.
Turnitin podudaranje [%]:	11

Ovom izjavom izjavljujem da je rad pod nazivom**: Angular aplikacija stranica za upravljanje građevinskom tvrtkom**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Ivica Lukić

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Marko Ćosić

SADRŽAJ

1. Uvod

U posljednje vrijeme, sve je jasnije kako je programiranje sve traženiji posao. Od prve pojave strojnoga koda dosta se toga promijenilo pa se danas može vidjeti koliko različitih programskih jezika postoji, te koliko je programiranje sveprisutno. Jedna od popularnijih grana programiranja zasigurno je web development, koji podrazumijeva izradu web stranice za Internet (World Wide Web) ili za Intranet (privatne mreže). Web development varira od razvoja jednostavne statičke stranice ili običnog teksta do kompleksnih web aplikacija, elektroničko poslovanje i društvenih servisa. U današnje vrijeme, kada je konkurencija u svim poslovnim segmentima velika, kada dobra promocija na društvenim i drugim mrežama može biti od velikog značaja, ne preostaje ništa drugo nego uložiti u kvalitetnu prezentaciju svoga rada. Točnije, izrada kvalitetne web aplikacije bila bi od velikog značaja i posljednji korak za uspješno poslovanje tvrtke. U tu svrhu definirana je i tema ovog završnog rada.

Ovaj završni rad strukturiran je u pet poglavlja. Drugo poglavlje predstavlja pregled područja teme gdje je dodatno navedeno pet sličnih rješenja. U trećem poglavlju teoretski su pokrivene tehnologije korištene u završnome radu, a u četvrtome poglavlju prikazana je instalacija potrebnih alata za izradu završnog rada, kao i primjerci izrade Angular komponenti. U posljednjem, petome poglavlju, dan je zaključak o izradi završnog rada.

1.1 Zadatak završnog rada

Zadatak ovog završnog rada je izrada Angular aplikacije stranice za upravljanje građevinskom tvrtkom. Aplikacija treba omogućiti upravljanje građevinskom tvrtkom. Od vođenja djelatnika, opreme, aktivnih projekata i slično. Za svaki projekt treba voditi djelatnike na projektu, potrebne resurse i opremu.

2. Pregled područja teme

Svjesni činjenice da je digitalizacija svuda oko nas, tako je i digitalizacija poslovanja postala uobičajena pojava danas. Razne tvrtke kao ključan čimbenik za uspješnost njihova poslovanja

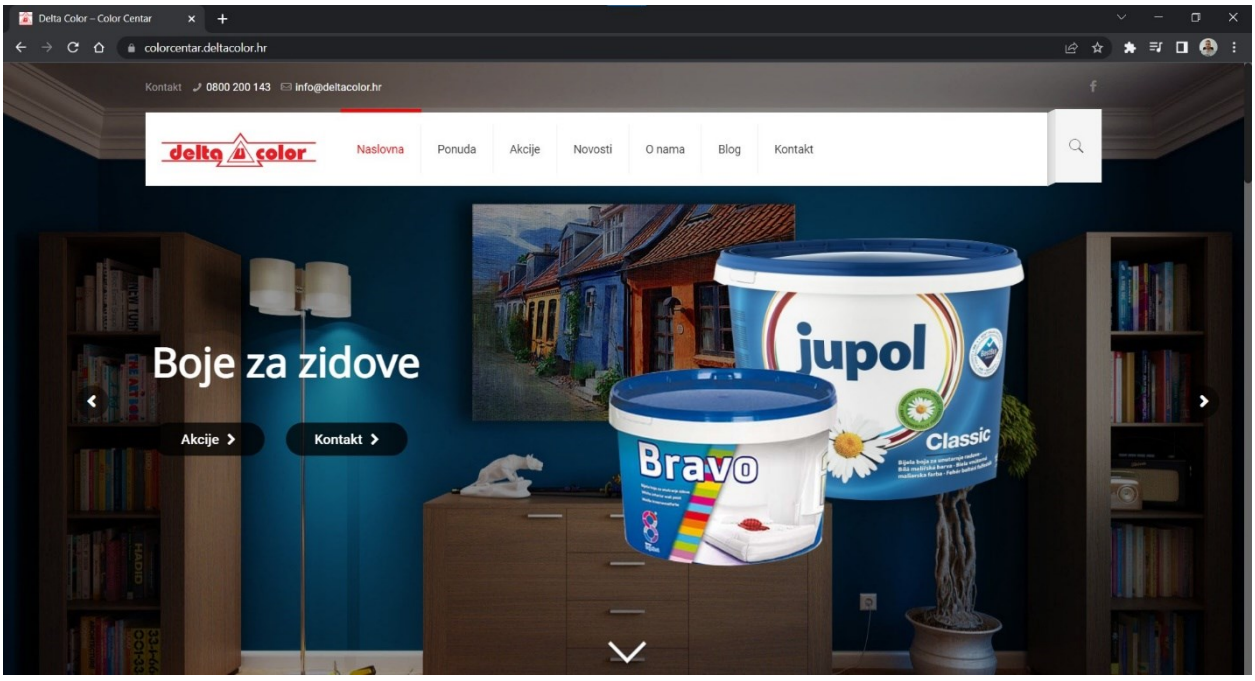
ističu ulaganjem u razvoj kvalitetnih web aplikacija. Web aplikacije predstavljaju vrlo učinkovit i informativan sadržaj, a tako je i u konkretnom primjeru za građevinsku tvrtku. Građevinska tvrtka Notus d.o.o. iz Slavonskog Broda do sada nije imala web aplikaciju, te se odlučila za izradu jedne kako bi potencijalnim novim klijentima, kupcima i partnerima na učinkovit i kreativan način prikazala dosadašnje, ali i buduće projekte. Osim prikaza vlastitih projekata, web aplikacije mogu poslužiti i za ostvarivanje kontakata s potencijalnim novim kupcima, ali i za internacionalno predstavljanje, što bi rezultiralo novim područjem rada.

U nastavku se mogu vidjeti primjeri web aplikacija za tvrtke koje su se odlučile na isti korak, koje su uložile u izradu web aplikacija kako bi poboljšale vlastito poslovanje. Navedene tvrtke bave se uslugama projektiranja, nabavnom građevinskih, sanitarnih i elektro materijala, hidro i termo izolacijama krovova te proizvodnjom PVC stolarije.

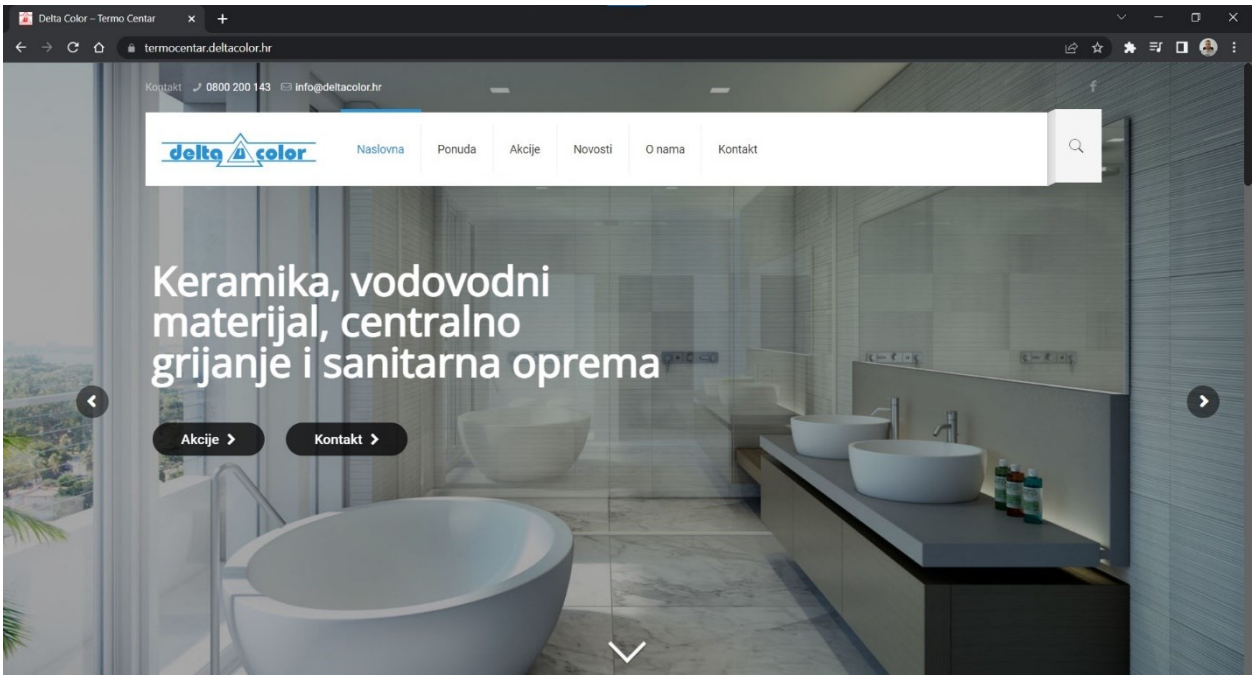
Primjeri takvih web aplikacija su: Constructo d.o.o. [1], tvrtka je koja se bavi projektantskim, savjetodavnim i konzultantskim uslugama, od projektne dokumentacije, dozvola za građenje, troškovnika pa sve do trgovanja nekretninama. DeltaColor [2], tvrtka je koja se sastoji od dvije poslovne grane, a to su Color Centar i Termo Centar. Color Centar bavi se opskrbom alata, boja, lakova, tapeta, žbuke, fasade i materijala za suhu gradnju. Termo Centar bavi se opskrbom keramike, vodovodnih materijala, centralnog grijanja i sanitarne opreme. Kristić izolacija [3], tvrtka koja je specijalizirana za sve vrste radova u području hidroizolacija, pri čemu ističu tri osnovne skupine usluga koje nude, a to su: ravni krovovi, podrumi i terase/balkoni. Metalka centar [4], tvrtka je koja nudi robu za kućanstvo, ali i za poduzetnike i obrtnike. U asortimanu nude: elektromaterijal, rasvjetu, bijelu tehniku, okove, alate, vijčanu robu, boje, aluminijske ograde, komarnike, prozorske klupice, opremu za grijanje i slično. Đaković montaža [5], tvrtka je koja se bavi uslugama montaže PVC, aluminijske i drvo-aluminijske stolarije, te garažnih vrata.

Izgled web aplikacija koje su prethodno navedene može se vidjeti u nastavku.

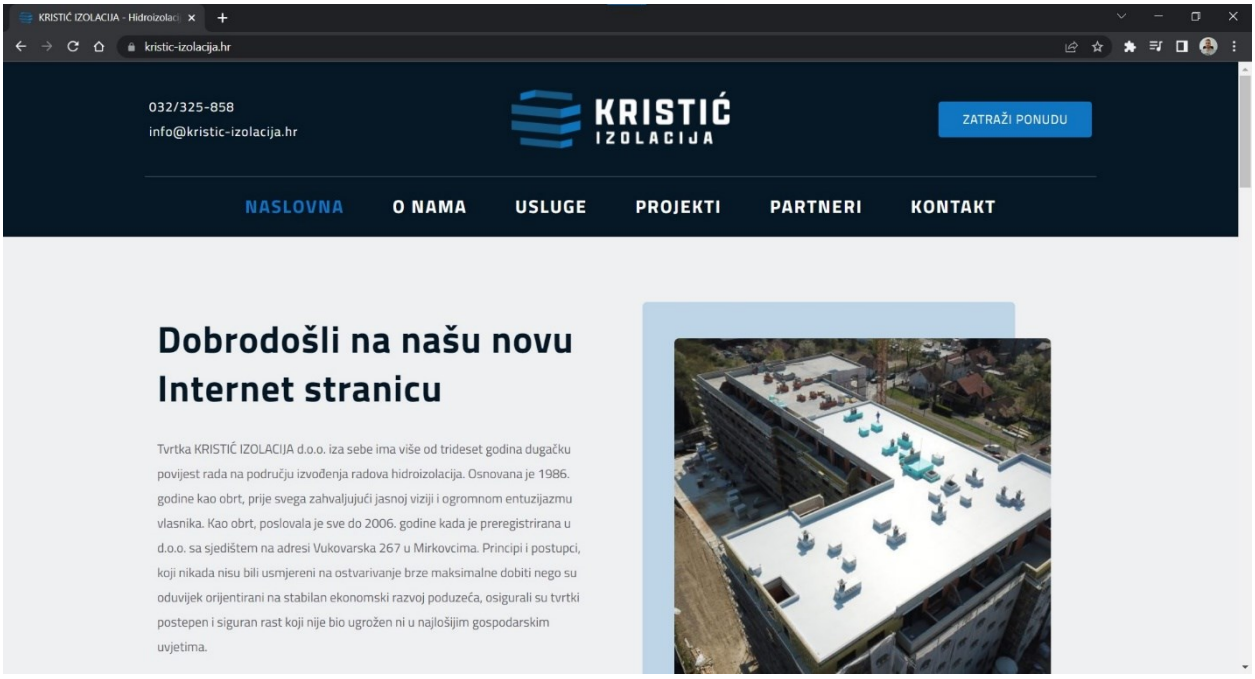
Slika 2.1. Izgled web aplikacije Constructo d.o.o.



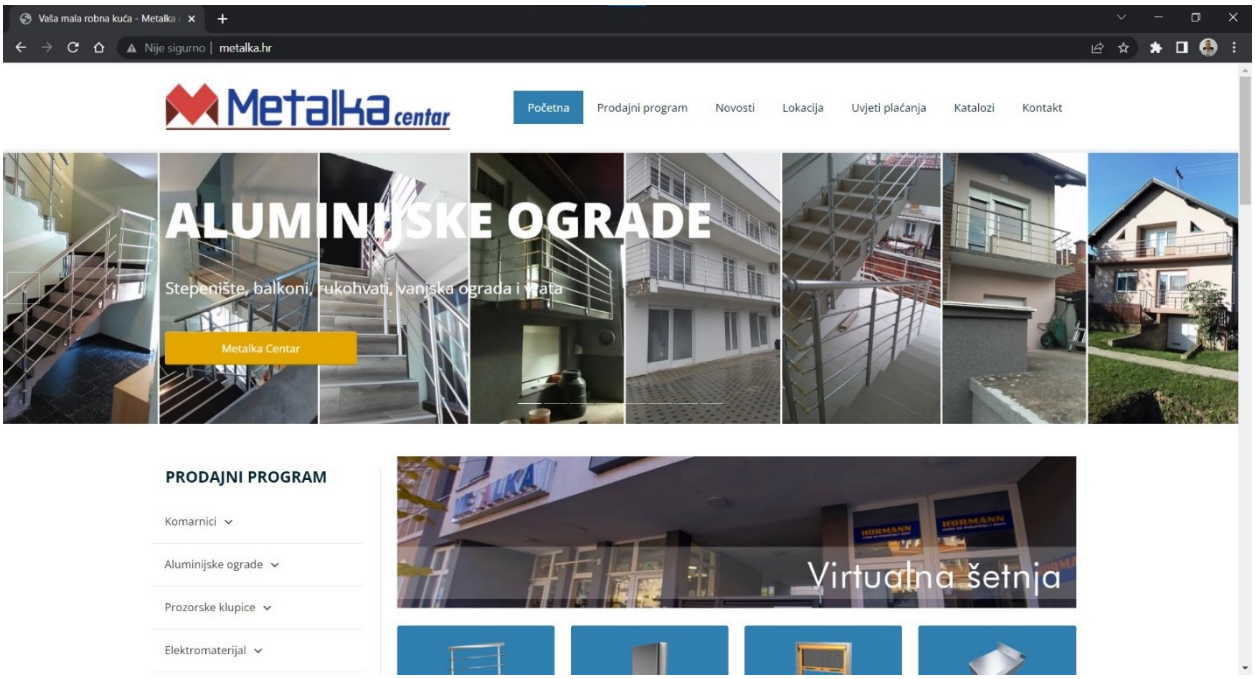
Slika 2.2. Izgled web aplikacije DeltaColor – Color Centar



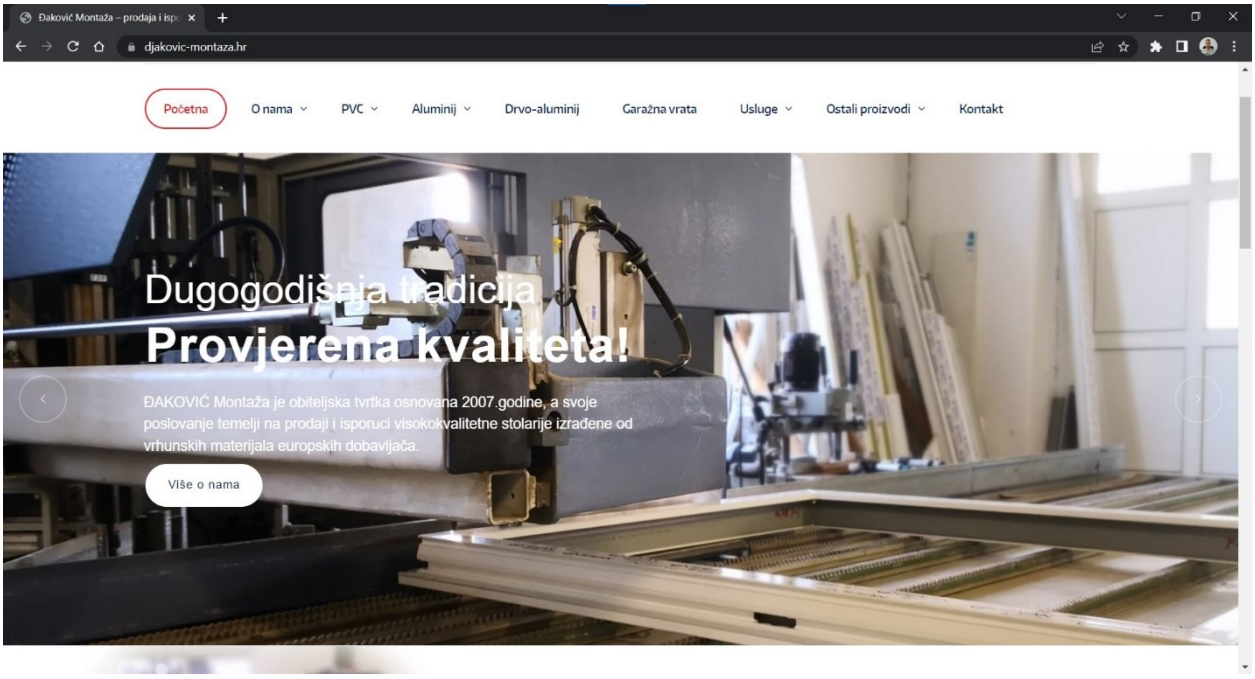
Slika 2.3. Izgled web aplikacije DeltaColor – Termo Centar



Slika 2.4. Izgled web aplikacije Kristić izolacija



Slika 2.5. Izgled web aplikacije Metalka centar



Slika 2.6. Izgled web aplikacije Đaković montaža

3. Tehnologije

U ovome će poglavlju biti predstavljene tehnologije koje su korištene za izradu završnog rada.

3.1 HTML

Osnovni jezik koji se koristi za izradu web aplikacija je HTML (engl. *HyperText Markup Language*). HTML predstavlja prezentacijski jezik za izradu web aplikacija. HTML jezik koristi se za postavljanje sadržaja i stvaranje svih hiperveza hipertekst dokumenta. Predstavlja jezik koji se lako upotrebljava, koji se lako uči, zbog čega je opće prihvaćen i popularan. Svoju sveprisutnost može pripisati jednostavnosti i tome što se od početka koristi kao besplatan te tako dostupan svima. Prikaz hipertekst dokumenta omogućuje web preglednik. Temeljna zadaća koju je HTML jezik morao ostvariti je uputiti web preglednik za pravilno prikazivanje hipertekstnog dokumenta. Pri tome treba obratiti pažnju da navedeni dokument izgleda isto bez obzira o kojem web pregledniku, operacijskom sustavu i računalu je riječ. Također, treba dodati i da HTML nije programski jezik, niti su ljudi koji ga koriste programeri. HTML ne može obaviti čak niti najjednostavnije matematičke operacije zbrajanja ili oduzimanja dvaju cijelih brojeva. Služi samo za opis hipertekst dokumenta. Aktualna inačica HTML-a koja se danas koristi je HTML5.

Svaki se HTML dokument sastoji od nekoliko osnovnih građevnih blokova – HTML oznaka (engl. *tag*). Svaki HTML element sastoji se od para HTML oznaka (engl. *tag*). Isto tako, svaki element može imati i atribute pomoću kojih se definiraju svojstva elementa. Na početku HTML dokumenta preporuča se postavljanje <!DOCTYPE> elementa kojim se označava DTD (engl. *Document Type Declaration*), čime se definira točna inačica standarda korištena za izradu HTML dokumenta. Nakon <!DOCTYPE> elementa, pomoću <html> elementa označava se početak HTML dokumenta. Unutar <html> elementa nalaze se i <head> element te <body> element. <head> element služi za postavljanje zaglavlja HTML dokumenta u kojemu se najčešće specificiraju jezične značajke HTML dokumenta kao i naslov (engl. *title*) stranice. Korištenjem određenih HTML elemenata u zaglavlju, dodaju se i stilska obilježja stranice, bila ona dodana kao referenca na vanjsku CSS datoteku ili direktno ugrađena (engl. *embedded*). Vrlo često unutar zaglavlja još se definiraju i skripte kreirane u JavaScript jeziku. Unutar <body> elementa kreira se sadržaj HTML dokumenta, točnije stranice koju reprezentira.

Svaka HTML oznaka (koja u paru kreira HTML element) počinje znakom < (manje od), a završava znakom > (više od). Zatvarajuća HTML oznaka kreira se slično kao otvarajuća, ali se nakon znaka < (manje od) dodaje kosa crta (engl. *slash*). Primjerice, <*primjer_oznake*> </*primjer_oznake*>.

U HTML-u moguće je koristiti i komentare. Unose se bilo gdje unutar HTML dokumenta i taj tekst neće biti prikazan na stranici, odnosno moći će se vidjeti samo ako je skripta otvorena s uređivačem koda (engl. *code editor*). Na ovaj način moguće je skrenuti pozornost na određeni dio skripte. Također, pomoću komentara je moguće isključiti dio skripte, ali sačuvati kod tog dijela. Komentari se otvaraju s oznakom <!--, a zatvaraju s oznakom -->.

Naslovi u HTML dokumentu oblikuju se radi uočljivosti i kako bi bili jedinstveni za cijelu web aplikaciju. Postoji šest veličina naslova. Početna oznaka najvećeg naslova je <h1>, a završna </h1>. Najmanji naslov počinje s <h6>, a završava s </h6>.

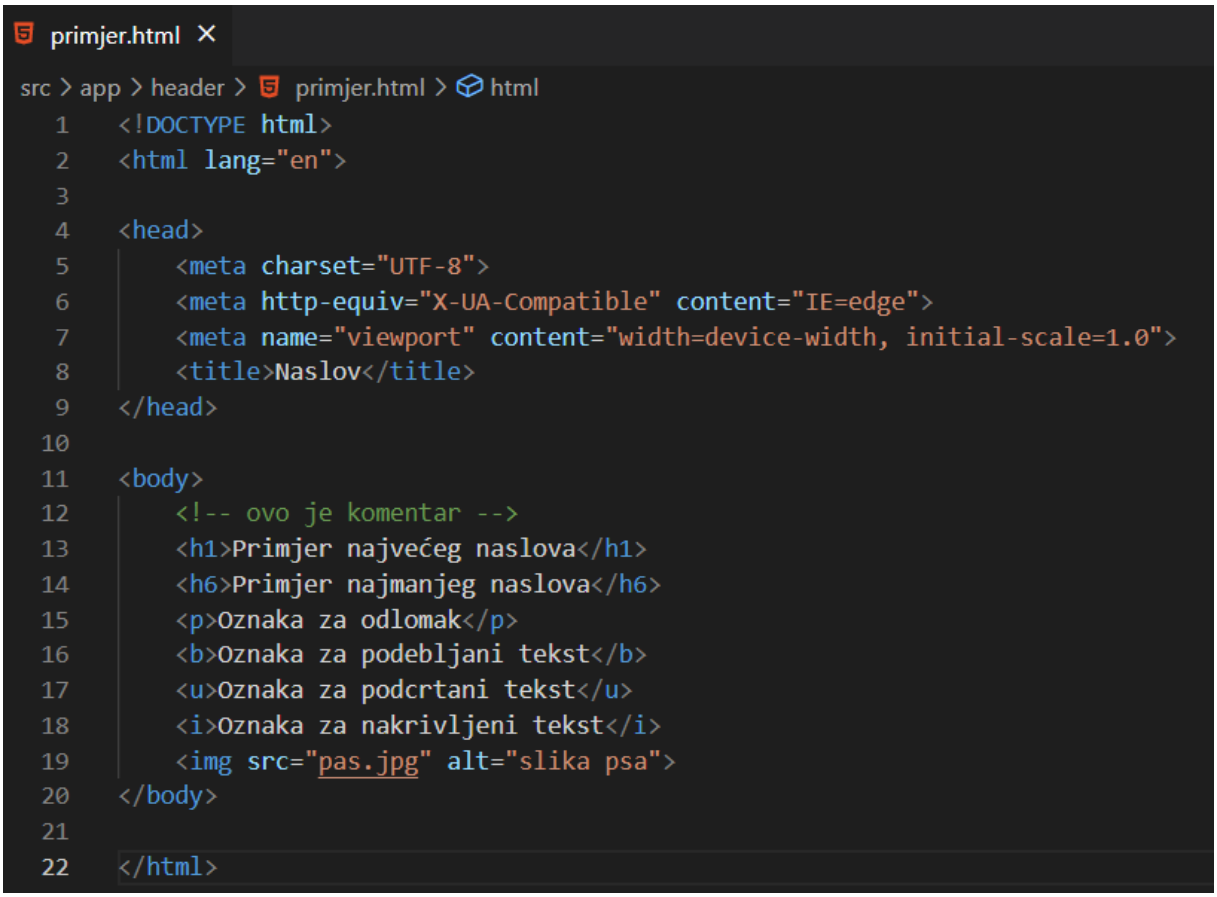
Osnovno postavljanje teksta unutar HTML dokumenta moguće je ostvariti postavljanjem određenih oznaka na početku teksta koji se želi oblikovati, te postavljanjem završne oznake na kraju teksta. Primjerice, predstavlja oznaku za odlomak (engl. *paragraph*), predstavlja oznaku za podebljani tekst (engl. *bold*), <u></u> predstavlja oznaku za podcrtani tekst (engl. *underlined*), predstavlja oznaku za nakrivljeni tekst (engl. *italic*).

Slike je moguće dodati oznakom <img>. Budući da navedena oznaka zahtijeva obilježje *src* (engl. *source*), tada se navedena oznaka promatra kao cjelovita i nema završnu oznaku. Opcionalno, može se dodati *alt* oznaka, koja može prikazati opis slike u slučaju njena neispravnog učitavanja.

Meta oznake dijelovi su HTML-a u stanici koje koriste tražilice da bi zapisale informacije o stranici. Ove oznake sadrže ključne riječi, naziv stranice, informacije o vlasništvu, opis i slično. Oni su među mnogim stvarima koje ispituju tražilice kada *gledaju* stranicu. Iako nije nužno, vrlo ih je korisno upotrebljavati.

Više informacija o HTML-u, kao i službena dokumentacija za ovu tehnologiju, može se pronaći na službenoj web stranici [6].

Do sada navedene oznake samo su neke od brojnih HTML oznaka. Primjer vezan za do sada spomenute oznake moguće je vidjeti u nastavku.



Slika 3.1. Primjer jednostavnog HTML dokumenta

3.2 CSS

CSS (engl. *Cascading Style Sheets*) je stilski jezik koji se koriti za opisivanje prezentacije dokumenta napisanog pomoću HTML jezika. S razvojem weba, prvotno su u HTML dodavani elementi za definiciju prezentacije, ali je vrlo brzo primijećena potreba za stilskim jezikom koji će HTML osloboditi potrebe prikazivanja sadržaja, što je i prvenstvena namjena HTML-a, kao i njegova oblikovanja, čemu danas i služi CSS. Drugim riječima, stil definira kako prikazati HTML elemente. Pomoću CSS-a se uređuje i sam izgled i raspored stranice. Aktualna inačica CSS-a koja se danas koristi je CSS3.

Stilska lista (engl. *style sheet*) u CSS-u sastoji se od nekoliko pravila. Svako se pravilo sastoji od selektora i deklaracijskog bloka.

Selektor (engl. *selector*) je dio označnog jezika (engl. *markup*) na koji se primjenjuje stil. Selektor može biti: 1) svi elementi istog tipa, primjerice svi h2 elementi, 2) elementi određenog id ili class atributa, pri čemu id predstavlja jedinstven element, a class može obuhvaćati više od jednog

elementa, 3) elementi u odnosu na druge elemente u DOM-u (*Document Object Modelu*). Pseudoklase su klase koje omogućuju opisivanje informacija koje nisu dostupne u DOM-u poput *:hover* koji identificira sadržaj samo ako korisnik drži pokazivač nad sadržajem.

Deklaracijski blok predstavljaju vitičaste zagrade unutar kojih se nalaze deklaracije. Svaka se deklaracija sastoji od svojstva, dvotočke (:) i vrijednosti. Između svake dvije uzastopne deklaracije mora se nalaziti točka zarez (;). Vrijednosti mogu biti ključne riječi poput *center* (sredina) i *inherit* (naslijedi), brojčane vrijednosti poput 100 (debljina fonta), 200px (200 piksela), 50vw (50% širine viewporta) ili 80% (80% širine prozora). Vrijednosti boja mogu biti ključne riječi, primjerice *red* za crveno, heksadecimalne vrijednosti, primjerice *#FF0000*, RGB vrijednosti od 0 do 255, primjerice *rgb(255, 0, 0)*, RGBA vrijednosti koje uključuju i aplha prozirnost, primjerice *rgba(255, 0, 0, 0.8)*.

CSS se može pisati unutar same HTML stranice na dva načina: 1) kao stilove u zaglavlju HTML dokumenta, između <stlye> i </style> elementa, 2) unutar samih HTML oznaka, primjerice * Primjer teskta *. Osim toga, CSS je moguće definirati u posebnom dokumentu i rabiti pomoću poziva: *<link rel="stylesheet" href="primjer.css">*.

Više informacija o CSS-u, kao i službena dokumentacija za ovu tehnologiju, može se pronaći na službenoj web stranici [7].

Primjer posljednje opisanog načina, točnije kada je CSS odvojen u poseban dokument, nalazi se u nastavku.

primjer.css

src > app > header > primjer.css > .img-container img

```
1   .img-container {
2       width: 500px;
3       height: 400px;
4   }
5
6   .img-container img {
7       width: 100%;
8       height: 100%;
9   }
```

Slika 3.2. Primjer jednostavnog CSS dokumenta

3.3 Bootstrap

Bootstrap je jedan od najpopularnijih okvira (engl. *framework*) za HTML, CSS i JavaScript. Besplatan je i otvorenog koda, a namijenjen je za frontend. Služi za razvoj responzivnih web

dokumenata, pri čemu je naglasak na mobilne uređaje (engl. *mobile-first*). Uključuje predloške zasnovane na HTML i CSS za tipografiju, obrasce, gumbe, tablice, navigaciju i još dosta kontrola kao i JavaScript dodatke.

Brojne su prednosti korištenja Bootstrap-a, a neke koje se ističu su: 1) lagana upotreba – bilo tko s osnovnim znanjem iz HTML i CSS web tehnologija može početi koristiti Bootstrap, 2) responzivni dizajn – mogućnost prilagodbe prikaza za različite vrste uređaja (pametni telefoni, tableti, računala), 3) pristup *mobile-first* – pristup razvoju web dokumenata prvo-za-mobilne uređaje, 4) kompatibilan s većinom preglednika – Chrome, Firefox, Internet Explorer, Safari i Opera.

Bootstrap ima vrlo dobro razvijen sustav mreže. Mreža je u grafičkom dizajnu skup vertikalnih i/ili horizontalnih linija koje omogućuju strukturiranje sadržaja. Mreže se koriste prilikom dizajniranja rasporeda i strukture sadržaja kod print dizajna. U web dizajnu mreža je vrlo efektivna metoda za izradu konzistentnog rasporeda elemenata brzi i efikasno koristeći HTML i CSS. Bootstrap uključuje sustav mreže koja je prilagođena mobilnim uređajima i vrlo responzivna. Bootstrapova se mreža proteže i do 12 stupaca, što ovisi o veličini ekrana uređaja.

Važno je za spomenuti i to kako Bootstrap dolazi i sa nekoliko JavaScript komponenti koje ne zahtijevaju druge biblioteke, kao primjerice jQuery. Omogućuju dodatno korisničko sučelje elemenata kao što su dijaloški okviri (engl. *dialog boxes*), opisi alata (engl. *tooltips*), trake napretka (engl. *progress bars*), te navigacijski padajući izbornici (engl. *navigation drop-downs*). Svaka se Bootstrap komponenta sastoji od strukture HTML-a, CSS deklaracija i, u određenim slučajevima, pratećeg JavaScript koda. Također, proširuju funkcionalnosti postojećih elemenata sučelja, uključujući primjerice funkciju automatskog dovršetka (engl. *auto-complete function*) za popunjavanje polja.

Više informacija o Bootstrap-u, kao i službena dokumentacija za ovu tehnologiju, može se pronaći na službenoj web stranici [8].

3.4 JavaScript

JavaScript, ili skraćeno JS, programski je jezik i jedna od temeljnih tehnologija World Wide Web-a uz HTML i CSS. Uzimajući u obzir 2022. godinu, 98% web aplikacija koriste JavScript na klijentskoj strani za ponašanje web aplikacija, često inkorporirajući druge biblioteke. Svi veći web pretraživači imaju JavaScript *engine* koji izvodi kod na klijentskim uređajima. Uključujemo ga u

web aplikaciju da bi je učinili dinamičnijom. JavaScript nam omogućuje definiranje ponašanja za prethodno napisane HTML elemente stranice.

JavaScript je objektno zasnovan skriptni jezik jer programer ne definira samo tip podataka, nego definira i vrstu funkcija koje se primjenjuju na strukture podataka. Na ovaj način struktura podataka postaje objekt koji uključuje i funkcije i podatke. Osim toga, programeri mogu kreirati i odnose između jednog i drugog objekta. JavaScript je skriptni jezik jer se sastoji od niza naredbi koje se izvode u *interpreteru*, a da se prethodno ne kompajlira sadržaj. Odnosno, ne prevodi se u strojni jezik iz kojega nikada nećemo saznati originalni jezik, nego se naredbe izravno čitaju iz koda. Zbog ove se karakteristike JavaScript izvršava na strani korisnika, točnije na uređaju na kojem je pokrenut sadržaj s JavaScriptom.

Prilagođen je ECMAScript standardu, a ima dinamičko tipkanje, objektnu orijentaciju na prototipu i prvoklasne funkcije. Također, podržava više paradigmi upravljanih događajima, te funkcionalne i imperativne stilove programiranja. Ima sučelja za programiranje aplikacija (engl. *API – Application Programming Interface*) za rad s tekstom, datumima, standardnim strukturama podataka i DOM-om (engl. *DOM – Document Object Model*). ECMAScript standard ne uključuje nikakav ulaz/izlaz (engl. *I/O*) podataka, kao što su umrežavanje, pohrana ili grafički sadržaji. U praksi, web preglednik ili drugi *runtime* sustav pruža JavaScriptu API-je za I/O. JavaScript *engine* ispočetka su samo korišteni u web preglednicima, a sada su ključne komponente nekih servera i mnoštva aplikacija. Najpopularniji *runtime* sustav za ovu upotrebu je Node.js.

Česta greška koja se može čuti jest da su Java i JavaScript ista stvar. Iako imaju slično ime, sintaksu i standardne biblioteke, Java i JavaScript nisu isti i uvelike se razlikuju po dizajnu.

U nastavku su navedene karakteristike JavaScripta:

- skriptni jezik – kao što je već spomenuto, JavaScript je lagani skriptni jezik napravljen za izvršavanje na strani klijenta u web pregledniku. Budući da nije dizajniran kao jezik opće namjene i posebno je dizajniran za web aplikacije, skup biblioteka također je usmjeren primarno ka web aplikacijama.
- baziran na interpreteru – JavaScript interpreterski je jezik, a ne kompajlerski, stoga je srodniji jezicima kao što su Ruby ili Python. Web pretraživač interpretira izvorni kod JavaScripta, liniju po liniju i izvodi ga. S druge strane, kompajlirani jezik mora biti kompajliran u izvršni byte-kod, kao što je primjerice slučaj kod Jave i C++.

- lagan za upotrebu – JavaScript nije kompajlirani jezik pa ne prevodi unaprijed u byte-kod. Kako god, prati kompilacijsku paradigmu koja se zove *just-in-time (JIT)*, što znači da se pretvori u bajt-kod prije samog pokretanja, a to ga čini lakim za upotrebu i ne tako memorijski zahtjevnim. Zbog ove karakteristike i slabiji uređaji mogu pokrenuti JavaScript.
- osjetljivost na velika i mala slova – JavaScript vrlo je osjetljiv na velika i mala slova. Sve ključne riječi, varijable, nazivi funkcija i drugi identifikatori mogu i moraju pratiti samo dosljedno pisanje velikih i malih slova. Primjerice, ako imamo *var hitCounter = 5* i *var hitcounter = 5*, varijable *hitCounter* i *hitcounter* su dvije potpuno različite varijable zbog razlike u nazivima. Također, važno je napomenuti da su i ključne riječi poput *var* isto osjetljive na velika i mala slova.

Više informacija o JavaScriptu-u, kao i službena dokumentacija za ovu tehnologiju, može se pronaći na službenoj web stranici [9].

3.5 TypeScript

TypeScript je besplatni programski jezik otvorenog koda kojeg je razvio i koji održava Microsoft. Strogi je sintaktički nadskup JavaScripta i omogućuje dodatno statičko pisanje u jeziku. Dizajniran je za razvoj velikih aplikacija i prevodi se u JavaScript. Budući da je nadskup JavaScripta, postojeći JavaScript programi također su važeći TypeScript programi.

TypeScript se može koristiti za razvoj JavaScript aplikacija kako za izvođenje na klijentskoj strani, tako i za izvođenje na poslužiteljskoj strani (kao što je slučaj s Node.js-om ili Denom). Dostupno je više opcija za transpilaciju. Može se koristiti zadani kompajler TypeScripta ili se može pozvati Babel kompajler za pretvaranje TypeScripta u JavaScript.

TypeScript podržava definiranje datoteka koje mogu sadržavati informacije o vrsti postojećih JavaScript biblioteka, slično kao što C++ datoteke zaglavlja mogu opisati strukturu postojećih objektnih datoteka. To omogućuje drugim programima da koriste vrijednosti definirane u datotekama kao da su statički upisani TypeScript entiteti. Postoje datoteke zaglavlja trećih strana za popularne biblioteke kao što su jQuery, MongoDB i D3.js. Dostupna su i TypeScript zaglavlja za osnovne module Node.js-a, što omogućuje razvoj Node.js programa unutar TypeScript-a.

Sam kompajler TypeScripta napisan je u TypeScriptu i prevodi se u JavaScript. Licenciran je pod licencom *Apache 2.0*. TypeScript je uključen kao prvoklasni programski jezik u Microsoft Visual Studio 2013, u ažuriranju broj 2 i kasnijim, zajedno s programskim jezikom C# i drugim Microsoft

programskim jezicima. Službeno proširenje također omogućuje Visual Studio 2012 da podržava TypeScript.

Više informacija o TypeScriptu-u, kao i službena dokumentacija za ovu tehnologiju, može se pronaći na službenoj web stranici [10].

3.6 Node.js

Node.js je backend okruženje JavaScipta, otvorenog koda, koje se izvodi na V8 *engine*-u, te izvodi JavaScript kod izvan web pretraživača. Kreiran je za izvođenje skalabilnih web aplikacija. Node.js omogućuje programerima korištenje JavaScripta za pisanje alata komadne linije i za skriptiranje serverske strane, odnosno za izvođenje skripti na serverskoj strani za kreiranje sadržaja dinamične web aplikacije prije nego što je stranica poslana na klijentski web pretraživač. Node.js zastupa *JavaScript bilo gdje* paradigmu, objedinjujući razvoj web aplikacija oko jednog programskog jezika, umjesto korištenja različitih programskih jezika za skripte na serverskoj strani i skripte na klijentskoj strani.

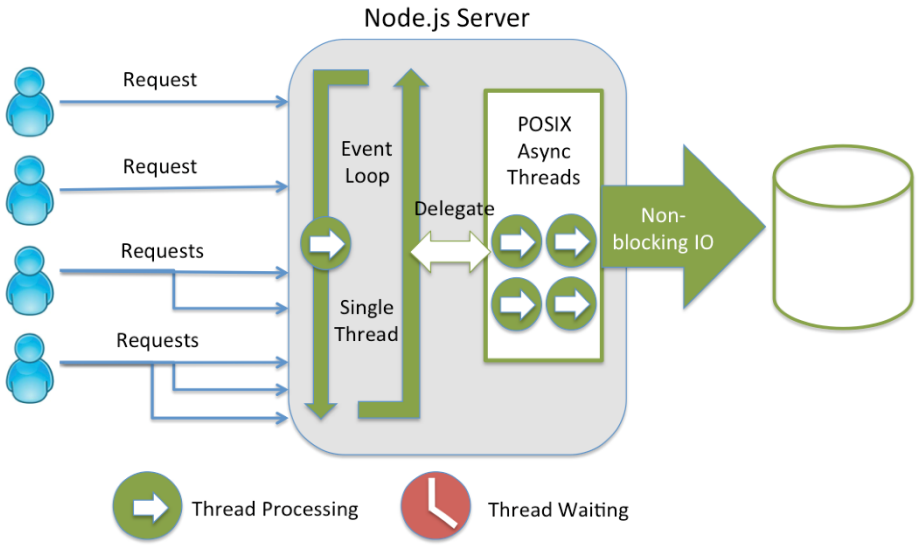
Node.js ima arhitekturu vođenu događajima sposobnu za asinkroni ulaz/izlaz (engl. *I/O*). Ovi izbori dizajna ciljaju na optimizaciju propusnosti i skalabilnosti u web aplikacijama s brojnim I/O operacijama, kao i za web aplikacije u stvarnom vremenu, primjerice programi za komunikaciju u stvarnom vremenu i igre preglednika.

Node.js omogućuje kreiranje web servera i mrežnih alata koristeći JavaScript i kolekciju modula koji upravljaju različitim ključnim funkcionalnostima. Moduli su dostupni za datotečni I/O sustav, mrežu (DNS, HTTP, TCP, TLS/SSL, ili UDP), binarne sustave, kriptografske funkcije, te druge funkcionalnosti. Moduli Node.js-a koriste API dizajniran da smanji kompleksnost pisanja serverskih aplikacija. JavaScript jedini je jezik kojeg Node.js podržava nativno, ali postoje i mnogi *compile-to-JS* jezici. Kao rezultat toga, Node.js aplikacije mogu biti pisane u jezicima kao što su CoffeScript, Dart, TypeScript, ClojureScript i slično.

Node.js primarno se koristi za izgradnju mrežnih programa kao što su web serveri. Najvažnija razlika između Node.js-a i PHP-a je ta što većina funkcija u PHP-u blokira do završetka (naredbe se izvršavaju tek nakon završetka prethodnih naredbi), dok Node.js funkcije ne blokiraju (naredbe se izvršavaju istovremeno ili čak paralelno i koriste povratne pozive za signaliziranje završetka ili neuspjeha).

Node.js službeno je podržan na Linux, macOS i Windows operacijskim sustavima.

Više informacija o Node.js-u, kao i službena dokumentacija za ovu tehnologiju, može se pronaći na službenoj web stranici [11].



Slika 3.3. Obrada klijentskih zahtjeva pomoću Node.js servera

3.6.1 Node Package Manager (NPM)

Node Package Manager, ili skraćeno NPM, je upravitelj paketa za programski jezik JavaScript. NPM je zadani upravitelj paketa za JavaScript *runtime* okruženje Node.js-a. Sastoji se od komandne linije klijenta, koja se još naziva NPM, i online baze podataka javnih i plaćenih privatnih paketa, koje se nazivaju NPM registar. Registru se pristupa putem klijenta, a dostupni se paketi mogu pregledavati i pretraživati putem NPM web stranice.

NPM je uključen kao preporučena karakteristika u Node.js instalaciji. Sastoji se od komandne linije klijenta koja komunicira s udaljenim registrom. Dopušta korisnicima korištenje i distribuciju JavaScript modula koji su dostupni u registru. Paketi u registru su u formatu CommonJS i uključuju datoteku meta podataka u JSON formatu. Više od 1.3 milijuna paketa dostupni su u glavnom NPM registru. Registar nema postupak provjere za podnošenje, što znači da paketi koji se tamo nalaze mogu potencijalno biti lošije kvalitete, nesigurni ili zlonamjerni. Umjesto toga, NPM se oslanja na izvješća korisnika kako bi uklonio pakete ukoliko krše pravila ako su niske kvalitete, nesigurni, ili pak zlonamjerni. NPM izlaže statistiku koja uključuje broj preuzimanja i broj zavisnih paketa kako bi pomogao programerima u procjeni kvalitete paketa.

NPM može upravljati paketima koji su lokalne ovisnosti određenog projekta, kao i globalno instaliranim JavaScript alatima. Kada se koristi kao upravitelj ovisnosti za lokalni projekt, NPM može instalirati u jednoj naredbi sve ovisnosti projekta kroz datoteku *package.json*. U

package.json datoteci svaka ovisnost može specificirati niz odgovarajućih verzija koristeći semantičku shemu verzioniranja, što omogućuje programerima da automatski ažuriraju svoje pakete dok istovremeno izbjegavaju neželjene promjene koje bi mogle izazvati pogreške.

3.7 Angular

Angular je okvir (engl. *framework*) otvorenog koda baziran na TypeScriptu koji se primjenjuje u web aplikacijama, a omogućuje kreiranje reaktivnih aplikacija na jednoj stranici (engl. *SPA – Single-Page-Application*) [12]. Vođen je od strane Angular tima u Google-u, te zajednicom pojedinaca i korporacija. Angular je potpuno ponovljena verzija istoga tima koji je kreirao AngularJS. Neke od značajki Angulara su povezivanje podataka, vlastiti sustav za kreiranje i upravljanje komponentama, podrška za upravljanje i validaciju formi, korištenje direktiva i slično. Dopušta brzo i kvalitetno razvijanje klijentskog dijela aplikacije, odnosno dijela kojeg korisnik vidi. Angular pojednostavljuje razvoj aplikacija tako što daje jednu razinu apstrakcije programeru, ali to može utjecati na fleksibilnost aplikacije.

Ponovljena verzija AngularJS-a zvala se *Angular2*, što je dovodilo do zbunjivanja kod programera. Angular tim najavio je promjenu kojom je pojasnio stvari, pri čemu se *AngularJS* (*Angular1*) odnosi na prvu verziju popularnog *framework*-a, a samo *Angular* na drugu verziju i sve verzije nakon nje. Do sada smo se susreli s dvanaest različitih verzija, od kojih je posljednja verzija 14, puštena u rad u lipnju 2022. godine. Razlog zašto imamo toliko puno verzija je taj što Angular tim u rad pušta novu verziju svakih šest mjeseci. To ne znači da svaka nova verzija ima velike razlike u odnosu na prethodnu, nego da se Angular tim dosljedno drži svojeg rasporeda objavljivanja novih verzija. Svaka je nova verzija kompatibilna s prethodnom (engl. *backward compatible*), primjerice verzija Angular11 kompatibilna je s verzijom Angular10 i slično. Najveća razlika je između verzije 1 i 2, stoga se verzija Angular1 naziva *AngularJS*, a verzija Angular2 i svaka nakon nje, jednostavno *Angular*.

3.7.1 Razlike između Angulara i AngularJS-a

Angular nema koncept *opsega* ili kontrolera, umjesto toga koristi hijerarhiju komponenti kao svoju primarnu arhitekturnu karakteristiku

Angular ima drugačiju sintaksu izraza, fokusira se na „ [ ] “ za povezivanje svojstva (engl. *property binding*) i „ ( ) “ za povezivanje događaja (engl. *event binding*)

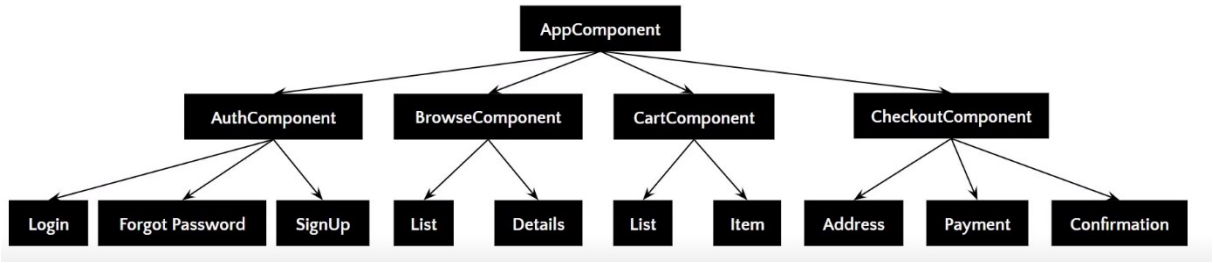
Modularnost – puno osnovnih funkcionalnosti prebačene su u module

Angular preporučuje korištenje Microsoftova TypeScript jezika, koji ima slijedeće karakteristike: statičko pisanje (koje uključuje generičko programiranje) i anotacije

Još neke razlike između Angulara i AngularJS-a koje se javljaju su: dinamičko učitavanje, asinkrone kompilacije predložaka, iterativni povratni pozivi omogućeni od strane RxJS-a, te podrška za pokretanje Angular aplikacija na poslužiteljima

3.7.2 Angular komponente

Angular komponente osnovni su gradivni elementi web aplikacije. Manje komponente svojom kompozicijom daju dobro strukturirani konačni proizvod, web aplikaciju, te u konačnici definiraju različite aspekte korisničkog sučelja.



Slika 3.4. Primjer kompozicije Angular komponenti koje u konačnici daju funkcionalnu web aplikaciju

Kao što je već spomenuto, komponente su gradivni elementi korisničkog sučelja Angular aplikacije. Komponente se povezuju s predlošcima i podskup su direktiva. Neke karakteristike komponenti su: 1) komponente su prilagođeni HTML elementi i svaki od tih elemenata može instancirati samo jednu komponentu, 2) koristi se TypeScript klasa za kreiranje komponente, te se u navedenu klasu dodaje „*@Component*“ dekorater, 3) dekorater prima objekt meta podataka koji daje informacije o komponenti, 4) komponenta mora pripadati u *NgModule* kako bi ju mogla koristiti druga komponenta ili aplikacija, 5) komponente kontroliraju svoje ponašanje tijekom izvođenja implementirajući *Life-Cycle* zakačke.

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'example';
}
```

Slika 3.5. Primjer koda koji se nalazi unutar AppComponent elementa

Kod koji je prikazan na slici broj 3.5. prikazuje kod elementa AppComponent, koji predstavlja čistu TypeScript klasu s „*@Component*“ dekoraterom. Objekt meta podataka pruža svojstva kao što su selektor, poveznica predloška (engl. *template url*), i slično. Poveznica predloška upućuje na HTML datoteku koja definira što ćemo vidjeti u web aplikaciji.

U datoteci *index.html* oznaka <app-root> odgovara selektoru komponente. Radeći to, Angular će ubrizgati odgovarajući predložak komponente.

```
<body>
  <app-root></app-root>
</body>
</html>
```

Slika 3.6. Ubrizgavanje <app-root> predloška komponente

3.7.3 Angular direktive

Angular direktive temeljni su razvojni koncepti. Direktive su u Angularu definirane kao klase koje mogu dodati novo ponašanje elementima u predlošku, ili promijeniti postojeće ponašanje. Svrha postojanja direktiva u Angularu je manevriranje DOM-om (engl. *Document Object Model*), što se ostvaruje dodavanjem novih elemenata u DOM, uklanjanjem elemenata ili mijenjanjem izgeda DOM elemenata.

Direktive su zamišljene kao funkcije koje se izvode nakon što se našu unutar DOM-a, a izvodi ih Angular kompajler kako bi proširio moć HTML-a s novom sintaksom. Ime direktiva može biti predefinirano, ili ih možemo imenovati po želji, što znači da se mogu imenovati na bilo koji način. Ovisno o predefiniranim direktivama, određuje se njihova upotreba – atribut, komentar, element, ili klasa.

Direktive unutar Angulara mogu biti kategorizirane u slijedeće tipove: direktiva komponenta, strukturna direktiva i atributna direktiva.

Komponenta direktiva posebne su direktive unutar Angulara koje se nazivaju komponentama jer ovaj tip direktive ima predložak ili poveznicu na predložak (engl. *template url*). Zapravo, to je komponenta direktiva koja pokazuje nešto u DOM-u.

Strukturne direktive koriste se kako bismo napravili promjene u rasporedu DOM-a. Elementi se mogu dodati ili ukloniti, čime se mijenja struktura DOM-a. Primjer je korištenje **ngIf*-a za dodavanje ili uklanjanje elemenata iz DOM-a, ili **ngFor*-a za navođenje elemenata svake iteracije.

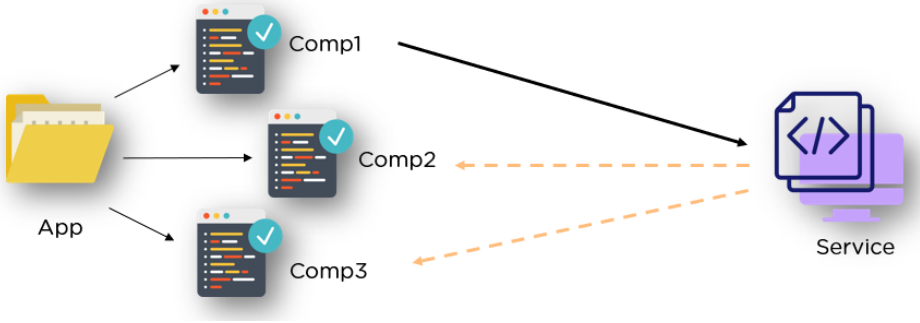
Atributne direktive koriste se kako bismo napravili ponašanje ili promjenu elementa u prikazu i ponašanju elementa. Primjerice, korištenje *ngStyle*-a za dodavanje stilova, ili *ngClass*-a za dodavanje CSS klasa.

3.7.4 Angular servisi i ubrizgavanje

Kao što i samo ime sugerira, Angular servisi pružaju *servise* raznim komponentama. Navedeni servisi variraju od jednostavnog unosa podataka do kompleksnih funkcionalnosti. Ranije u ovom radu upoznali smo se s Angular komponentama. Korisničko sučelje web aplikacije razvijeno je ugrađivanjem nekoliko komponenti u glavnu komponentu. Kako god, navedene komponente generalno se koriste samo za potrebe renderiranja. Koriste se samo za definiranje onoga što se pojavljuje u korisničkom sučelju. U idealnom slučaju, zadaci poput dohvaćanja podataka i slika, mrežnih povezivanja, te upravljanja bazama podataka se ne izvode. Postavlja se pitanje kako se rješavaju spomenuti zadaci i što ako više od jedne komponente obavljaju slične zadatke. Odgovor su servisi – oni brinu za spomenuto obavljajući sve potrebne operacije za komponente. Servisi sprječavaju prepisivanje (dupliciranje) koda, mogu biti napisani jednom i ubrizgani u komponente koje će ih koristiti. Servis može biti funkcija, varijabla, ili značajka koju web aplikacija treba.

Angular servisi su objekti koji se instanciraju jednom za vrijeme korištenja aplikacije. Sadrže metode koje održavaju podatke u jednom životnom ciklusu aplikacije, a podaci su dostupni cijelo

vrijeme. Glavni je zadatak servisa da organizira i podijeli poslovnu logiku, modele ili podatke i funkcije s drugim komponentama Angular aplikacije. Najčešće se implementiraju ubrizgavanjem ovisnosti.



Slika 3.7. Jednostavni primjer funkcioniranja Angular servisa

Servisi su u Angularu zapravo TypeScript klase koje imaju *@Injectable* dekorater. Ovaj dekorater govori Angularu da je klasa servis i da može biti ubrizgana u komponente koje trebaju taj servis. Također, mogu ubrizgati duge servise kao ovisnosti.

Kao što je već objašnjeno, servisi se koriste kako bi podijelili jedan (isti) dio koda u svim komponentama koje ga koriste. Servisi se koriste za očuvanje poslovne logike.

3.7.5 Angular usmjeravanje

Svatko je mogao primijetiti da se URL (engl. *Uniform Resource Locator*) web aplikacije promijeni svaki puta kada kliknemo na link korisničkoga sučelja. Promjena linka javlja se zato što navigiramo kroz web aplikaciju, a to se ostvaruje pomoću uobičajenog mehanizma koji se naziva usmjeravanje (engl. *routing*).

Angular usmjeravanje (engl. *routing*) ima jako važnu ulogu jer se u osnovi koristi za kreiranje jednostranih aplikacija (engl. *Single Page Applications*). Takve se aplikacije učitaju samo jednom, a novi se sadržaj dodaje dinamički. Aplikacije kao Google, Facebook, Twitter i Gmail samo su neki primjeri jednostranih aplikacija. Najveća prednost jednostranih aplikacija je ta što pružaju izvrsno korisničko iskustvo bez potrebe za čekanjem za učitavanje stranica, što čini jednostrane aplikacije vrlo brzima i reaktivnima. Usmjeravanje specificira navigaciju s kosom crtom, / (engl. *forward slash*), nakon koje ide put (engl. *path*) koji definira novi sadržaj.

3.7.6 Angular forme

Angular forme (engl. *forms*) sastavni su dio web aplikacija. Gotovo svaka web aplikacija ima formu koju korisnik treba popuniti. Angular forme služe za prijavu korisnika, ažuriranje profila,

unos osjetljivih informacija, ali i za obavljanje drugih zadataka vezanih za unos podataka. Razlikujemo dvije vrste Angular formi, a to su forme vođene predloškom (engl. *template-driven forms*) i reaktivne forme (engl. *reactive forms*).

U metodi formi vođenih predloškom koristi se konvencionalna oznaka za kreiranje formi. Angular automatski interpretira i koristi reprezentaciju objekta forme za oznaku. Kontrole se mogu dodati u formu korištenjem NGModel oznake. Više kontrola mogu se grupirati koištenjem NGControlGroup modula. Vrijednost forme može biti generirana korištenjem *form.value* objekta. Podaci forme se izvoze (engl. *export*) kao JSON vrijednosti nakon poziva *submit* metode. Osnovne HTML validacije se mogu koristiti kako bi validirali polja formi. U slučaju prilagođenih validacija, mogu se koristiti direktive. Ova je metoda ujedno i najlakši način za kreiranje Angular aplikacije.

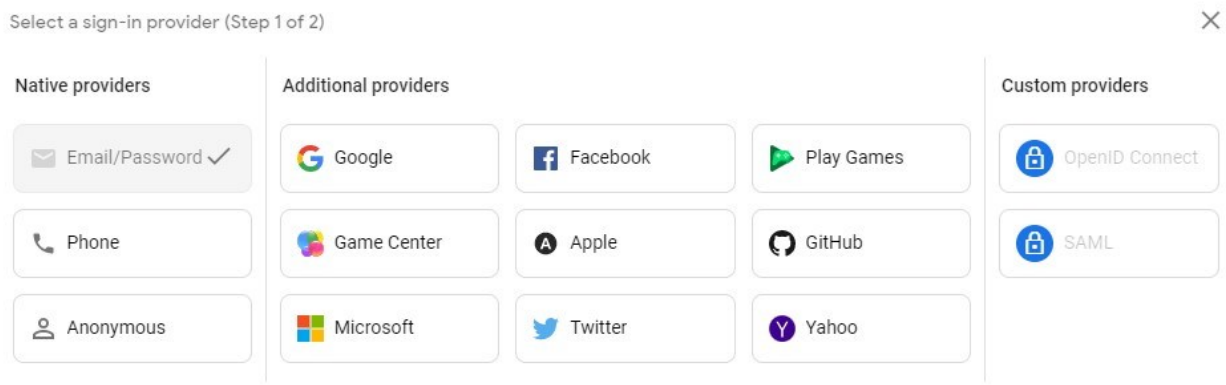
Drugi pristup koji koristi reaktivne forme je programska paradigma orijentirana oko toka podataka i propagacije promjena. S reaktivnim formama, komponenta direktno upravlja tokom podataka između kontrola forme i modela podataka. Reaktivne forme se kodiraju, za razliku od formi vođenih predloškom. Reaktivne forme krše tradicionalni deklaracijski pristup, te eliminiraju anti-obrazac ažuriranja podatkovnog modela putem dvosmjernog povezivanja podataka.

3.8 Google Firebase

Google Firebase je platforma za kreiranje web i mobilnih aplikacija [13]. Sadrži mnoštvo alata, a jedan od poznatijih definitivno je *Realtime Database*, točnije API koji sinkronizira podatke aplikacije na Android, iOS i web uređajima i sprema ih na Firebase-ov oblak. Ostali alati uvelike olakšavaju posao programerima jer imaju usluge koje bi programeri morali samostalno razvijati. Primjerice analitika, provjera autentičnosti, konfiguracija, usluge poslužitelja, pohrana podataka i slično. Svi navedeni servisi se nalaze u oblaku, što znači da imaju *backend* komponentu koja se u potpunosti održava i vodi od strane Google-a. Razlog je jednostavan, a to je olakšati cjelokupni razvojni proces programerima kako bi se mogli usredotočiti na zahtjeve korisnika i iskustvo aplikacije. Za razliku od dosad viđenih razvojnih procesa, koji podrazumijevaju razvoj i *frontend* i *backend* dijela aplikacije, Firebase platforma omogućuje drugačiji pristup i upravo je to jedna od brojnih prednosti njena korištenja. Osim što značajno smanjuje vrijeme razvijana za programere, skalira se po potrebi i besplatan je.

Za potrebe izrade ovog završnog rada korištene su dvije usluge Firebase platforme, a to su *Realtime Database* i *Authentication*. *Realtime Database* [14] predstavlja NoSQL bazu podataka koja ima drugačije funkcionalnosti i optimizacije u usporedbi s relacijskom bazom, a pružena je

iz oblaka. Neke od ključnih sposobnosti podrazumijevaju sinkronizaciju podataka umjesto slanja HTTP zahtjeva, očuvanje podataka pri gubitku veze zbog lokalnog spremanja podataka, pristup bazi podataka bez potrebe za aplikacijom poslužitelja, te sigurnost i ovjeravanje podataka. *Authentication* [15] je Firebase usluga koja nudi alate za razne funkcionalnosti, kao što su prijavljivanje i odjavljivanje korisnika, ali i općenito upravljanje korisnicima. Također, treba dodati da Firebase omogućuje i ugradnju vlastitog sustava provjere autentičnosti. U nastavku su dani primjeri oblika za prijavu korisnika, a za potrebe izrade završnog rada korištena je prijava putem email adrese, što se može vidjeti kao već odabrana opcija.



Slika 3.8. Oblici prijava za prijavu korisnika

4. Izrada aplikacije

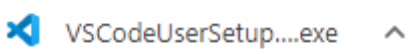
U ovom će poglavlju biti objašnjena izrada Angular web aplikacije za upravljanje građevinskom tvrtkom. Web aplikacija bit će i informativnog karaktera, gdje će se, osim osnovnih podataka o tvrtki i kontakt informacija, moći vidjeti prijašnji projekti tvrtke i budući projekti u sekciji novosti. Putem prijave korisnika, direktoru tvrtke omogućeno je dodavanje novosti, gdje će osim najavljivanja novih projekata, moći upravljati partnerima i kooperantima na projektima.

Prije svega, potrebno je instalirati alate koji se koriste za izradu projekta, a to su Visual Studio Code, Node.js i AngularCLI. Nakon toga, bit će potrebno kreirati novi projekt zajedno sa svim komponentama, servisima i direktivama koje će se koristiti za izradu web aplikacije. Nadalje, bit će dani primjeri koda koji prikazuju kako se web aplikacija razvija. U konačnici, bit će postavljen završni izgled web aplikacije.

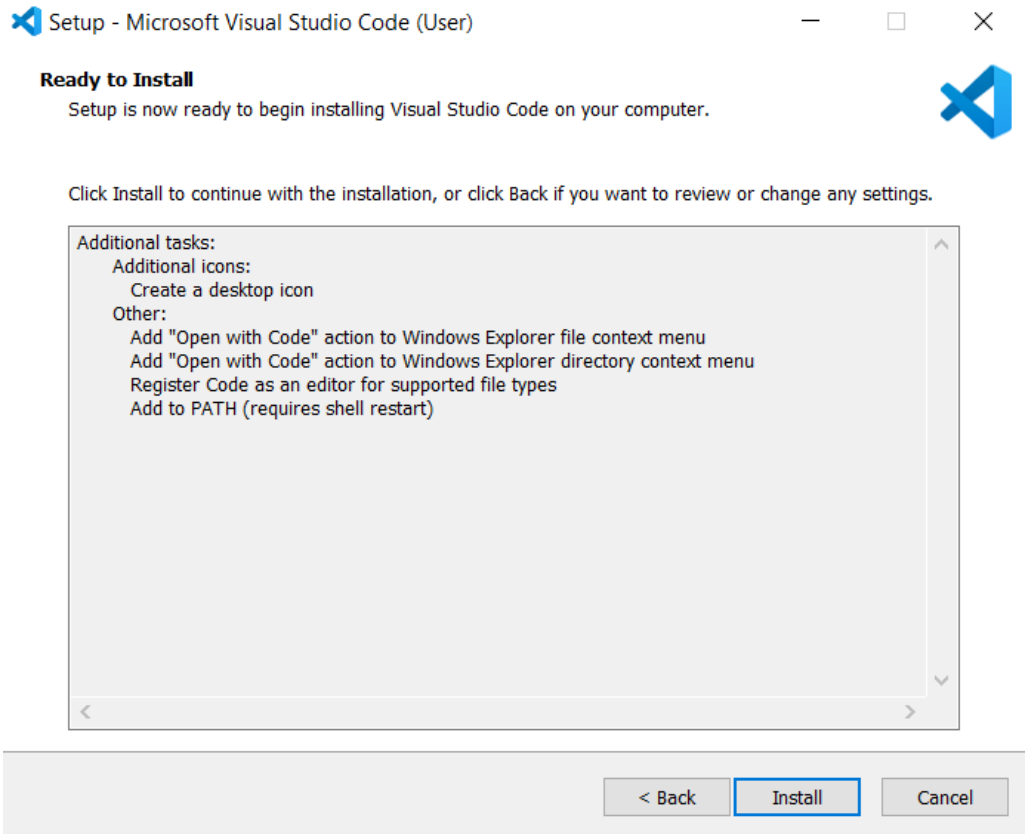
4.1 Instalacija alata

Prvo je potrebno odabrati uređivač koda (engl. *IDE – Integrated Development Environment*) po želji, u ovom slučaju je to Visual Studio Code. Riječ je o jednostavnom i moćnom uređivaču koda koji je dostupan za Linux, macOS i Windows operacijske sustave. Dolazi s ugrađenom podrškom za JavaScript, TypeScript i Node.js, a ima i bogat ekosistem ekstenzija za druge programske jezike, kao što su C++, C#, Java, Python, PHP, Go, .NET. Osim navedenih podrški za JavaScript i TypeScript, u izradi ovog završnog rada Visual Studio Code koristit će se i za pisanje HTML i CSS koda.

Odgovarajuća verzija instalacije koja je potrebna za naš operacijski sustav može se pronaći na službenoj Visual Studio stranici u rubrici preuzimanja. Nakon što je preuzimanje gotovo, možemo kliknuti na skinutu datoteku za početak instalacije.

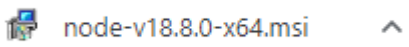


Slika 4.1. Skinuta instalacijska datoteka za Visual Studio Code

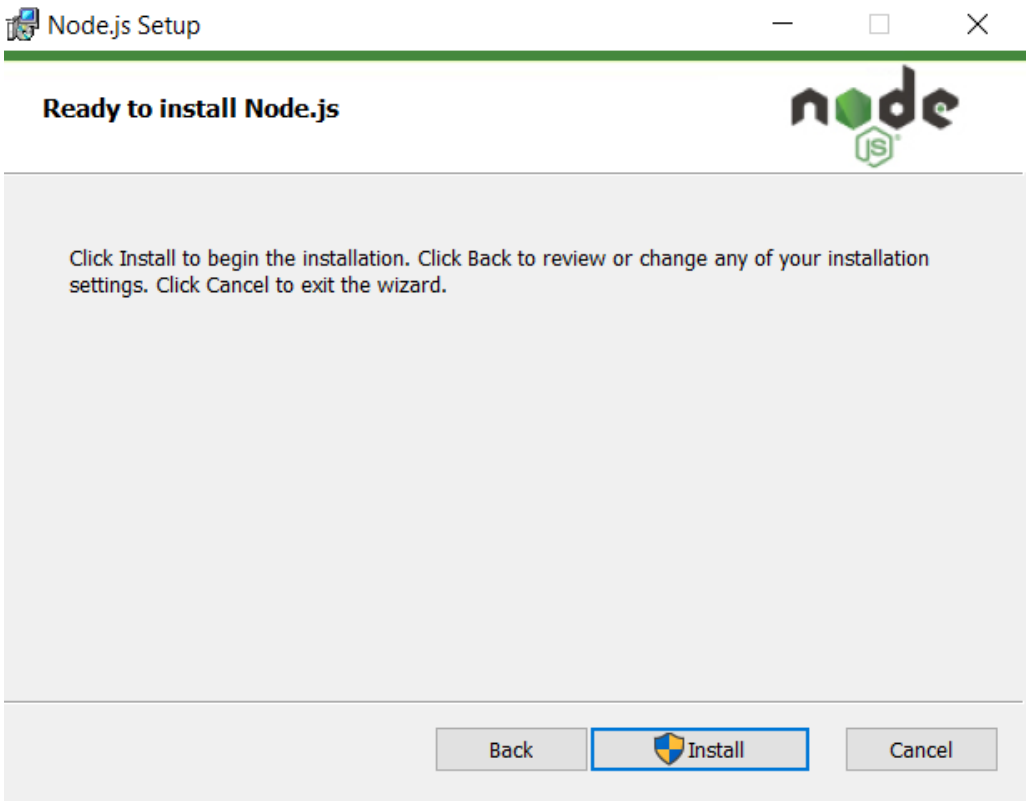


Slika 4.2. Instalacija Visual Studio Code uređivača koda

Slijedeće što je potrebno instalirati je Node.js koji služi kao JavaScript asinkrona platforma dizajnirana za stvaranje skalabilnih mrežnih aplikacija. Odgovarajuća verzija instalacije koja je potrebna za naš operacijski sustav može se pronaći na službenoj Node.js stranici u rubrici preuzimanja. Nakon što je preuzimanje gotovo, možemo kliknuti na skinutu datoteku za početak instalacije.



Slika 4.3. Skinuta instalacijska datoteka za Node.js

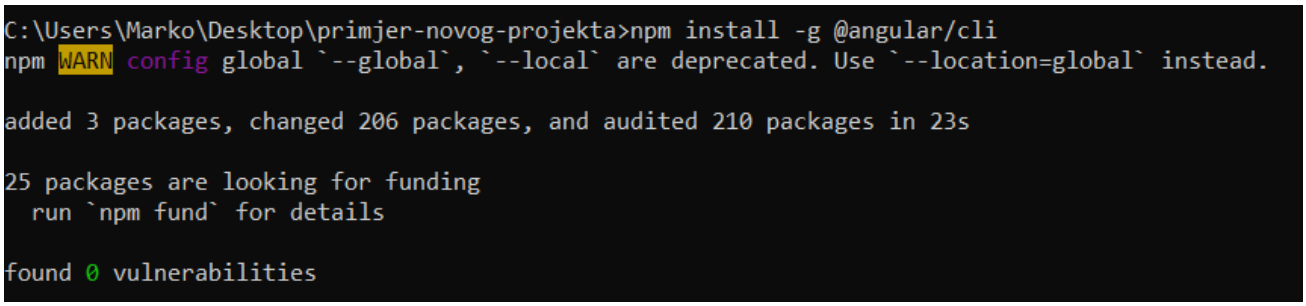


Slika 4.4. Instalacija Node.js-a

Na kraju, potrebno je instalirati Angular CLI (engl. *Command-Line Interface*). Angular CLI je alat sučelja naredbenog retka koji se koristi za inicijalizaciju, razvoj i održavanje Angular aplikacija direktno iz naredbene ljuske (engl. *command shell*).

Potrebno je kreirati folder unutar kojeg ćemo smjestiti novi projekt. Za primjer instalacije, kreiran je novi folder i bit će kreiran novi projekt. Prvo je potrebno pokrenuti naredbeni redak kao administrator. Nakon toga, unijeti naredbu *cd* kako bismo se smjestili u folder gdje ćemo kreirati novi projekt. U ovom slučaju naredba izgleda ovako: *C:\WINDOWS\system32>cd C:\Users\Marko\Desktop\primjer-novog-projekta*

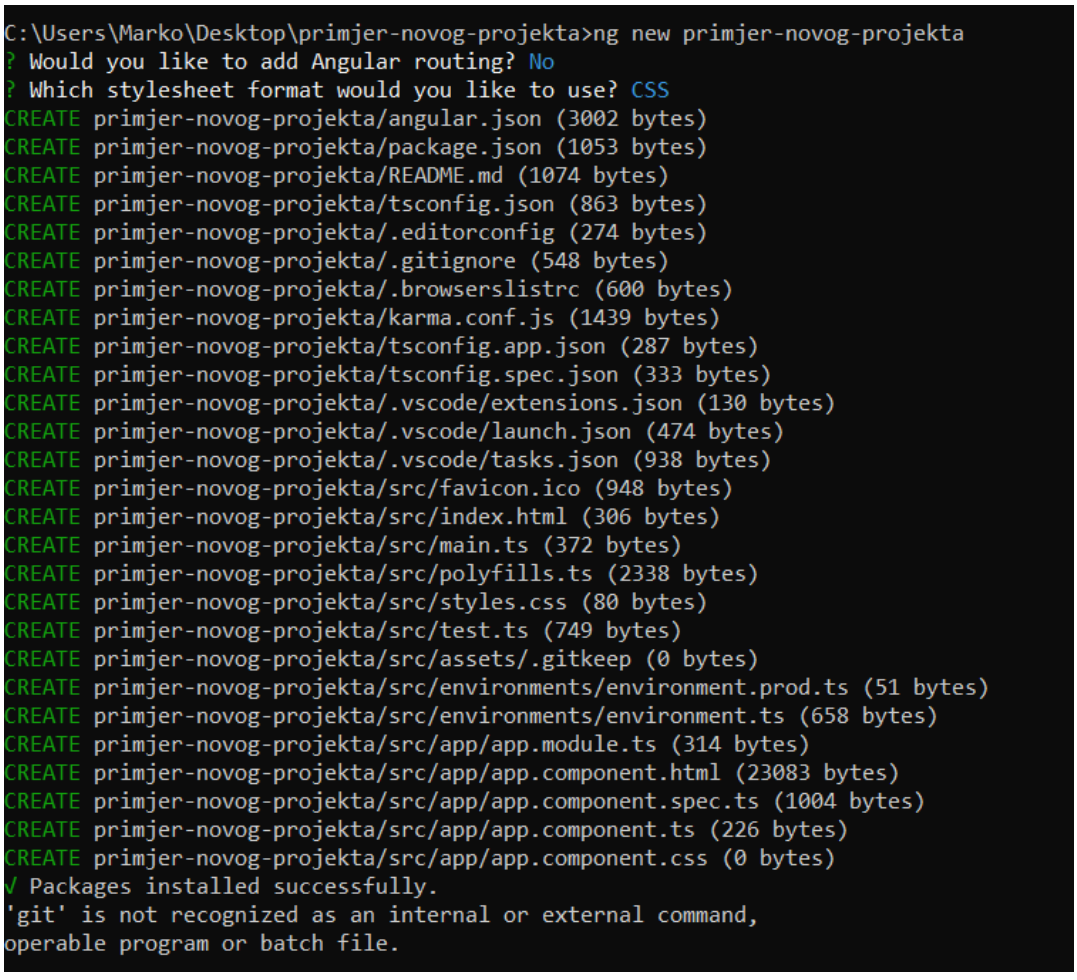
Nakon toga, potrebno je unijeti naredbu: *npm install –g @angular/cli*. Iz naredbe se može zaključiti da koristimo *node package manager (npm)*, te da instalaciju vršimo globalno *-g*. Opcionalno, u naredbu možemo dodati *@latest* pa bi naredba izgledala ovako: *npm install –g @angular/cli@latest*, što znači da instaliramo posljednju verziju Angulara.



Slika 4.5. Instalacija Angular CLI-a

4.2 Izrada novog projekta

Nakon što smo instalirali Angular CLI, potrebno je unijeti naredbu *ng new*, u našem slučaju ona izgleda ovako: *ng new primjer-novog-projekta*. Nakon pokretanja naredbe, dobijemo ovakav ispis:



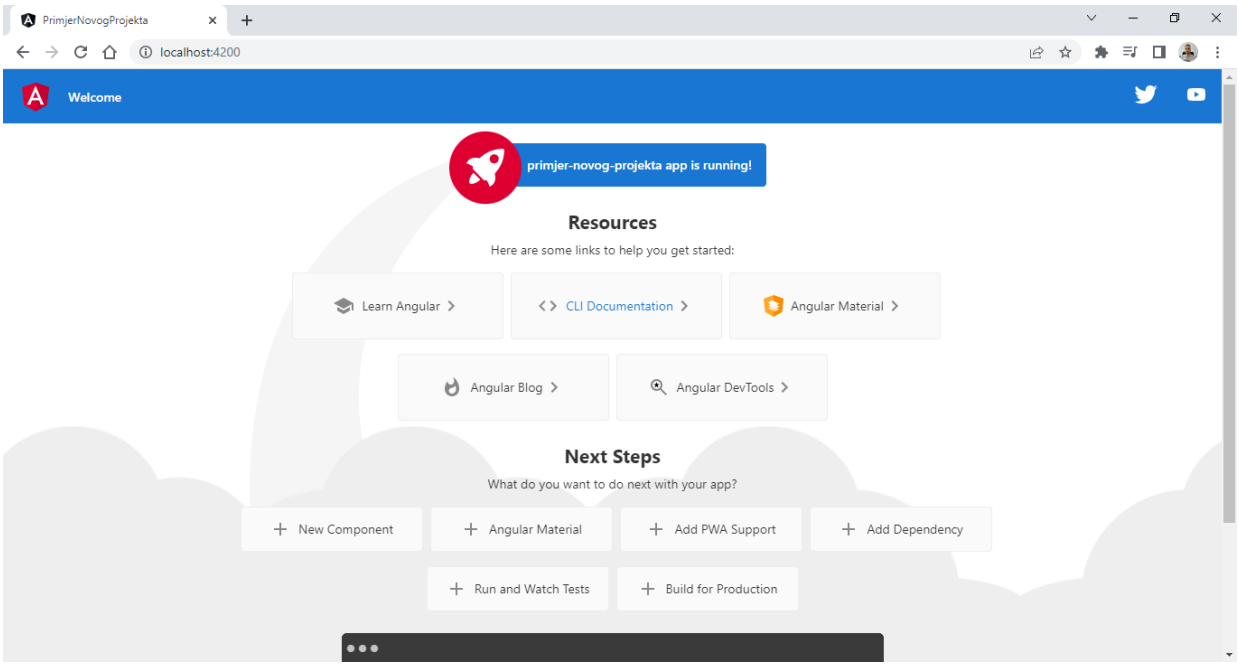
Slika 4.6. Kreiranje novog projekta

Posljednji korak je unos naredbe *ng serve*. Nakon što smo pozicionirani naredbom *cd* unutar foldera gdje je kreiran projekt, naredba *ng serve* pokreće našu aplikaciju na web lokaciji

http://localhost:4200/. Svaki put kada napravimo neku izmjenu unutar projekta i spremimo, naredba *ng serve* automatski će primijeniti te promjene na web aplikaciji. Treba imati na umu da jednom kada ju pokrenemo, naredba *ng serve* mora ostati pokrenuta dokle god radimo na projektu i želimo imati ovakav dinamički prikaz promjena na web aplikaciji.

```
C:\Users\Marko\Desktop\primjer-novog-projekta\primjer-novog-projekta>ng serve
√ Browser application bundle generation complete.

Initial Chunk Files   | Names         |  Raw Size
vendor.js             | vendor        |   1.78 MB |
polyfills.js          | polyfills     | 319.85 kB |
styles.css, styles.js | styles        | 211.88 kB |
main.js               | main          |  48.08 kB |
runtime.js            | runtime       |   6.54 kB |

                      | Initial Total |   2.35 MB

Build at: 2022-08-28T21:00:10.286Z - Hash: d05bd57a128faf45 - Time: 27429ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **


√ Compiled successfully.
```

Slika 4.7. Pokretanje naredbe *ng serve*



Slika 4.8. Početni izgled web aplikacije

4.3 Angular elementi unutar projekta

U ovom će poglavlju biti prikazani Angular elementi koji su kreirani i koji se koriste unutar projekta. Ranije u ovome radu isti su elementi objašnjeni teorijski, a sada će biti prikazan način na koji se kreiraju i kako izgledaju unutar projekta. Treba napomenuti da je zbog konvencije pisanja korištena engleska nomenklatura, primjerice za komponentu *projekti* dano je ime *projects*, i slično.

Glavne komponente koje treba istaknuti unutar projekta su: *contact, header, home, news, partners* i *projects*. Kreirana je i jedna *image* direktiva koja služi za dijaprojekciju slika. Kreirana su i dva servisa: *NewsService* i *ProjectsService*. U konačnici, kreirana je i *app-routing.module.ts* datoteka koja je korištena za usmjeravanje unutar web aplikacije.

4.3.1 Angular komponente

Angular komponente moguće je kreirati na dva načina; prvi je način *ručnim* kreiranjem i on zahtjeva više posla. Primjerice, desnim klikom unutar *app* foldera možemo odabrati kreiranje novog foldera (engl. *New Folder*) ili nove datoteke (engl. *New File*). Nakon odabira, možemo dodati novu datoteku. Također, moramo se pobrinuti da u *app.module.ts* datoteci dodamo novu komponentu koju smo stvorili unutar *NgModule* deklaracija i ažuriramo popis *import* komponenti na vrhu datoteke. Drugi način je da unosimo naredbe unutar terminala. Primjerice za kreiranje nove komponente unijet ćemo naredbu *ng generate component naziv-komponente* ili skraćeno *ng g c naziv-komponente*. Ovaj način jednostavniji je za korištenje, ali i bolji je odabir jer smo tako sigurni da nismo zaboravili niti jedan uvoz (engl. *import*) komponenti unutar *app.module.ts* datoteke, što moramo napraviti u prvoj opciji. Svi se uvozi komponenti obavljaju automatski.

Kao primjer kreiranja nove Angular komponente bit će korištena druga opcija.

```
PS C:\Users\Marko\Stranica\notus> ng generate component nova-komponenta
CREATE src/app/nova-komponenta/nova-komponenta.component.html (30 bytes)
CREATE src/app/nova-komponenta/nova-komponenta.component.spec.ts (656 bytes)
CREATE src/app/nova-komponenta/nova-komponenta.component.ts (310 bytes)
CREATE src/app/nova-komponenta/nova-komponenta.component.css (0 bytes)
UPDATE src/app/app.module.ts (3540 bytes)
```

Slika 4.9. Kreiranje nove Angular komponente

4.3.2 Angular direktive

Angular direktive moguće je, kao i komponente, kreirati na dva načina. Prvi način je *ručnim* dodavanjem nove direktive u željeni folder, pri čemu trebamo paziti da ažuriramo *app.module.ts* datoteku. Drugi način je i u ovome slučaju bolja opcija, a izvršava se naredbom *ng generate directive naziv-direktive* ili skraćeno *ng g d naziv-direktive*.

U nastavku će biti prikazan primjer direktive koja je kreirana u svrhu prikaza dijaprojekcije slika unutar projekta.

image.directive.ts

src > app > shared > image.directive.ts > ...

```typescript
import { Directive, HostListener, ElementRef } from '@angular/core';

@Directive({
  selector: '[appImage]'
})
export class ImageDirective {

  constructor(private element: ElementRef) { }

  @HostListener('click')

  imageChanged() {
    var src: any = this.element.nativeElement.src;
    var prev: any = document.getElementById("preview");
    prev.src = src;
    var imageSlide = document.getElementsByClassName("img-slide");
    for (let i = 0; i < imageSlide.length; i++) {
      imageSlide[i].classList.remove("active");
    }
    this.element.nativeElement.parentElement.classList.add("active");
  }

}
```

Slika 4.10. Kod *image.directive.ts* datoteke

4.3.3 Angular servisi

U ovome projektu, Angular servisi kreirani su na slijedeći način. Desni klik na željeni folder gdje želimo dodati servis, odabir opcije nova datoteka (engl. *New File*) i imenovanje *naziv-servisa.service.ts*. U projektu su, kao što je već spomenuto, kreirana dva servisa, a u nastavku se može vidjeti njihov izgled.

projects.service.ts | news.service.ts

src > app > projects > projects.service.ts > ProjectsService > addProjects

```ts
import { Projects } from "../shared/project.model";
import { Subject } from "rxjs";
import { Injectable } from "@angular/core";

@Injectable()

export class ProjectsService {
    projectsChanged = new Subject<Projects[]>();
    startedEditing = new Subject<number>();

    private projects: Projects[] = [
        new Projects('Projekt Bakaceva', 'Urbana vila', ''),
        new Projects('Projekt Muraviceva', 'Urbana vila', ''),
        new Projects('Projekt Gunduliceva', 'Stambeno poslovna zgrada', ''),
        new Projects('Projekt Teslina', 'Stambena zgrada', ''),
        new Projects('Projekt Strossmayerova', 'Stambena zgrada', '')
    ];

    getProjects() {
        return this.projects.slice();
    }

    getProject(index: number) {
        return this.projects[index];
    }

    addProject(project: Projects) {
        this.projects.push(project);
        this.projectsChanged.next(this.projects.slice());
    }

```

Slika 4.11. Kod *projects.service.ts* datoteke

projects.service.ts | news.service.ts

src > app > news > news.service.ts > NewsService

```ts
import { Injectable } from "@angular/core";
import { Subject } from "rxjs";
import { ProjectsService } from "../projects/projects.service";
import { Projects } from "../shared/project.model";
import { News } from "./news.model";

@Injectable()

export class NewsService {
    newsChanged = new Subject<News[]>();

    private news: News[] = [
        new News('Novi projekt br 1', 'Najavljujemo novi projekt za ljeto 2022. godine', 'https://www.nekretnine365
            new Projects('Novi projekt', 'stambeno poslovna zgrada', 'https://www.nekretnine365.com/files/05-2022/a
        ]),
        new News('Novi projekt br 2', 'Najavljujemo novi projekt za zimu 2021. godine', 'https://www.nekretnine365.
            new Projects('Novi projekt', 'urabana vila', 'https://www.nekretnine365.com/files/05-2022/ad416763/slav
        ])
    ];

    constructor(private projectsService: ProjectsService) { }

    getNews() {
        return this.news.slice();
    }

    getNewsElement(index: number) {
        return this.news[index];
    }
```

Slika 4.12. Kod *news.service.ts* datoteke

4.3.4 Angular usmjeravanje

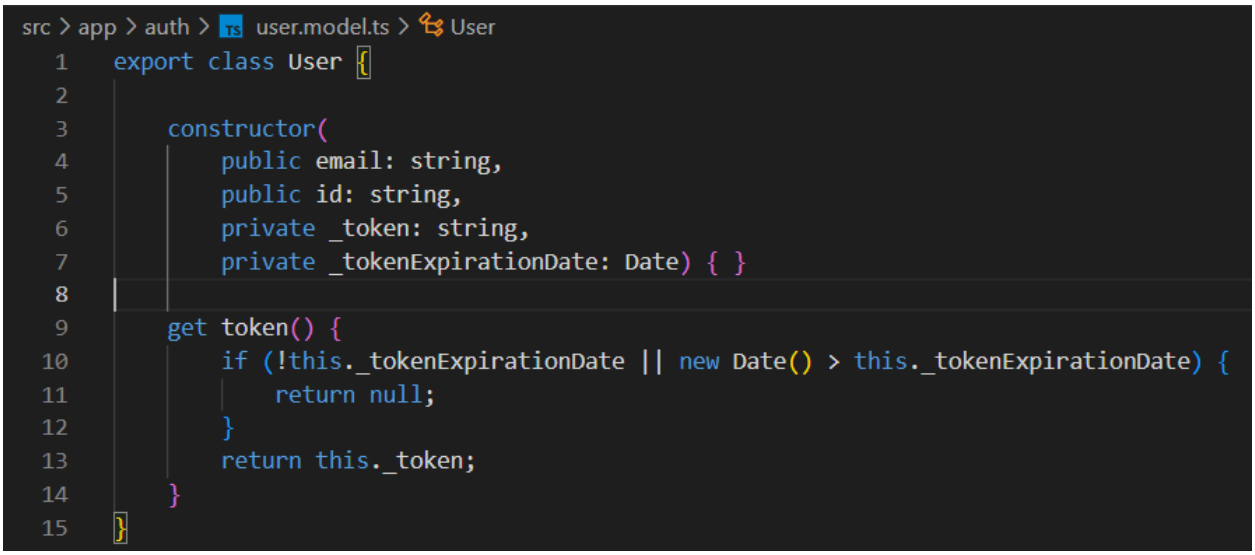
Kao i prethodno opisani Angular servisi, Angular usmjeravanje (engl. *routing*) kreirano je na isti način. Desni klik na željeni folder gdje želimo dodati datoteku, odabir opcije nova datoteka (engl. *New File*) i imenovanje *naziv.module.ts*. Odabran je naziv *app-routing.module.ts*, a u nastavku je prikazan dio koda navedene datoteke.

```
import { ProjectsDetailsComponent } from "./projects/projects-details/projects-details.component";
import { ProjectsStartComponent } from "./projects/projects-start/projects-start.component";
import { ProjectsComponent } from "./projects/projects.component";
import { UserLoginComponent } from "./user-login/user-login.component";

const appRoutes: Routes = [
    { path: '', redirectTo: '/home', pathMatch: 'full' },
    {
        path: 'news', component: NewsComponent, children: [
            { path: '', component: NewsStartComponent },
            { path: 'edit', component: NewsEditComponent },
            { path: ':id', component: NewsDetailsComponent },
            { path: ':id/edit', component: NewsEditComponent }
        ]
    },
    {
        path: 'projects', component: ProjectsComponent, children: [
            { path: '', component: ProjectsStartComponent },
            { path: ':id', component: ProjectsDetailsComponent },
        ]
    },
    { path: 'home', component: HomeComponent },
    { path: 'partners', component: PartnersComponent },
    { path: 'contact', component: ContactComponent },
    { path: 'userlogin', component: UserLoginComponent }
];

@NgModule({
    imports: [RouterModule.forRoot(appRoutes)],
    exports: [RouterModule]
})
```

Slika 4.13. Kod *app-routing.module.ts* datoteke

4.4 Firebase autorizacija

Kao posljednji korak prilikom izrade web aplikacije odrađena je autorizacija korisnika, pri čemu se koristi email adresa za prijavu. Neki drugi oblici koje je moguće koristiti za prijavu korisnika navedeni su u ranijim poglavljima. Kreiran je model korisnika *user* koji sadrži svojstva propisana od strane Firebase-a koje korisnik mora imati prilikom prijave, a to su: email adresa, id korisnika, token i token do isteka prijave. Kod koji se nalazi u modelu korisnika može se vidjeti u nastavku.



Slika 4.14. Kod *user.model.ts* datoteke

Iz priloženog se koda može vidjeti da je kreirana i funkcija za dohvaćanje (engl. *getter*), koja se naziva *token()* i njena je zadaća vraćanje tokena korisnika. Firebase ima određeno da token do isteka prijave vrijedi jedan sat, nakon čega je potrebno odraditi ponovnu prijavu. Zbog toga je kreirana navedena funkcija u slučaju da korisnikov token prije isteka prijave još uvijek vrijedi.

Nadalje, kreirana je *auth* komponenta pomoću koje ćemo riješiti problem prijave korisnika. Unutar *auth.component.html* dokumenta kreirana je Angular forma (engl. *form*) pomoću koje korisnik može unijeti email adresu i lozinku za prijavu. Također, dodana je obrada pogreške koja potencijalno može nastati prilikom prijave. Primjerice, pokušaj prijave neregistriranog korisnika, unos neispravne lozinke, unos neispravne email adrese i slično. Dodana je opcija prijave i registracije korisnika kako bi novi korisnik prvo mogao odraditi registraciju, a potom se prijaviti. U nastavku se može vidjeti izgled *auth.component.html* datoteke.

```
src > app > auth > auth.component.html > div.row > div.col-xs-12.col-md-6.col-md-offset-3 > div
Go to component
<div class="row">
    <div class="col-xs-12 col-md-6 col-md-offset-3">
        <div class="alert alert-danger" *ngIf="error">
            <p>{{ error }}</p>
        </div>
        <div *ngIf="isLoading" style="text-align: center;">
            <app-loading-spinner></app-loading-spinner>
        </div>
        <form #authForm="ngForm" (ngSubmit)="onSubmit(authForm)" *ngIf="!isLoading">
            <div class="form-group">
                <label for="email">Email adresa</label>
                <input type="email" id="email" class="form-control" ngModel name="email" required email>
            </div>
            <div class="form-group">
                <label for="password">Lozinka</label>
                <input type="password" id="password" class="form-control" ngModel name="password" required
                    minlength="6">
            </div>
            <div>
                <button class="btn btn-primary" type="submit" [disabled]="!authForm.valid">{{ isLoginMode ? 'Prijava' :
                    'Registracija' }}</button> |
                <button class="btn btn-primary" (click)="onSwitchMode()" type="button">Prebaci na {{ isLoginMode ?
                    'registraciju' : 'prijavu' }}</button>
            </div>
        </form>
    </div>
</div>
```

Slika 4.15. Kod *auth.component.html* datoteke

Kreirana je i funkcija *onSubmit(form)* koja se izvršava klikom na gumb i koja dohvaća podatke potrebne za prijavu ili registraciju korisnika. Izgled navedene funkcije može se vidjeti u nastavku.

```
src > app > auth > auth.component.ts > AuthComponent
23
24      onSubmit(form: NgForm) {
25          if (!form.valid) {
26              return;
27          }
28          const email = form.value.email;
29          const password = form.value.password;
30          let authObs: Observable<AuthResponseData>;
31          this.isLoading = true;
32
33          if (this.isLoginMode) {
34              authObs = this.authService.login(email, password);
35          } else {
36              authObs = this.authService.signUp(email, password);
37          }
38
39          authObs.subscribe(resData => {
40              console.log(resData);
41              this.isLoading = false;
42              this.router.navigate(['/home']);
43          }, errorMessage => {
44              console.log(errorMessage);
45              this.error = errorMessage;
46              this.isLoading = false;
47          });
48
49          form.reset();
50      }
```

Slika 4.16. Kod *onSubmit(form)* funkcije

Također, kreiran je i servis pod nazivom *auth.service.ts* kako bismo odvojili određene funkcije i tako dobili bolji pregled unutar samog koda. U navedenom se servisu nalaze funkcije za prijavu, odjavu i registraciju korisnika, ali i za automatsku prijavu i odjavu korisnika. Automatska prijava korisnika implementirana je u slučaju da korisnik, nakon što je prijavljen, osvježi stranicu. Tako ostaje prijavljen, u protivnome bi se ponovno morao prijavljivati. Automatska odjava korisnika implementirana je zbog ranije navedenog tokena i vremena do isteka prijave. Kao što je već spomenuto, Firebase omogućuje korisniku da ostane prijavljen jedan sat, a nakon toga token ističe. Funckija *autoLogout()* radi provjeru je li token istekao, te ako je, poziva funkciju *logout()* koja odjavljuje korisnika. Izgled navedenih funkcija može se vidjeti u nastavku.

```
27    signUp(email: string, password: string) {
28        return this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyAYQrq1akbfESA5dZUVMjefzfT
29            {
30                email: email,
31                password: password,
32                returnSecureToken: true
33            }
34        ).pipe(catchError(this.handleError), tap(resData => {
35            this.handleAuthentication(resData.email, resData.localId, resData.idToken, +resData.expiresIn);
36        }));
37    }
```

Slika 4.17. Kod *signUp(email, password)* funkcije

```
39    login(email: string, password: string) {
40        return this.http.post<AuthResponseData>('https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=AIzaSyAYQrq1akbfESA
41            {
42                email: email,
43                password: password,
44                returnSecureToken: true
45            }
46        ).pipe(catchError(this.handleError), tap(resData => {
47            this.handleAuthentication(resData.email, resData.localId, resData.idToken, +resData.expiresIn);
48        }));
49    }
50
51    autoLogin() {
52        const userData: {
53            email: string;
54            id: string;
55            _token: string;
56            _tokenExpirationDate: string;
57        } = JSON.parse(localStorage.getItem('userData'));
58        if (!userData) {
59            return;
60        }
61
62        const loadedUser = new User(userData.email, userData.id, userData._token, new Date(userData._tokenExpirationDate));
63
64        if (loadedUser.token) {
65            this.user.next(loadedUser);
66            const expirationDuration = new Date(userData._tokenExpirationDate).getTime() - new Date().getTime();
67            this.autoLogout(expirationDuration);
68        }
69    }
```

Slika 4.18. Kod *login(email, password)* i *autoLogin()* funkcija

```
71      logout() {
72          this.user.next(null);
73          this.router.navigate(['/auth']);
74          localStorage.removeItem('userData');
75          if (this.tokenExpirationTimer) {
76              clearTimeout(this.tokenExpirationTimer);
77          }
78          this.tokenExpirationTimer = null;
79      }
80
81      autoLogout(expirationDuration: number) {
82          this.tokenExpirationTimer = setTimeout(() => {
83              this.logout();
84          }, expirationDuration);
85      }
```

Slika 4.19. Kod *logout()* i *autoLogout()* funkcija

Autorizacijom korisnika ostvarili smo da neautorizirani korisnici, odnosno oni koji nisu prijavljeni, mogu samo vidjeti novosti, projekte i partnere. Autorizirani, odnosno prijavljeni korisnici mogu i uređivati novosti, projekte i partnere. Izgled navedenih stranica prikazan je u poglavlju *Konačan izgled web aplikacije*.

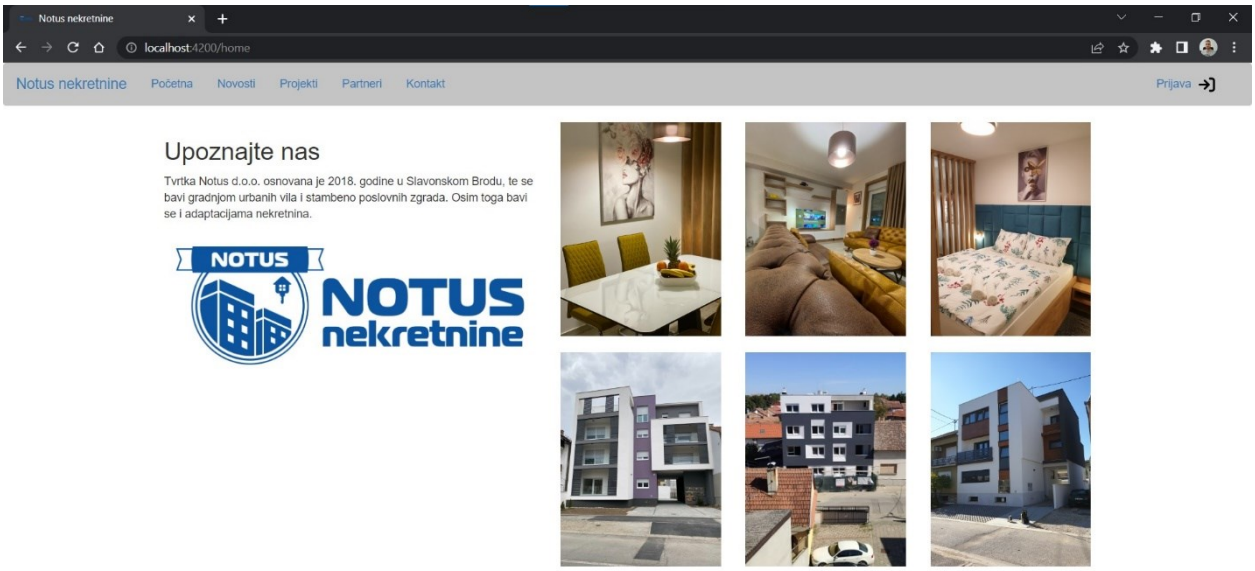
No postoji jedan problem koji treba riješiti. Primjerice, kako bi vidjeli partnere, može se i *ručno* unijeti */partners* unutar web pretraživača. Kako bi uredili partnere, može se *ručno* unijeti */partners/edit*. Opisano se ne smije dogoditi jer uređivanje partnera mora biti omogućeno samo za autorizirane korisnike. Stoga je potrebno kreirati *auth.guard.ts* komponentu koja će prilikom pokušaja *ručnog* unosa rute osigurati da su neautorizirani korisnici preusmjereni na autorizaciju. Izgled navedene komponente nalazi se u nastavku.

```
9   export class AuthGuard implements CanActivate {
10
11      constructor(private authService: AuthService, private router: Router) { }
12
13      canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean | UrlTree | Promise<boolean | UrlTree> | Observable<boolean | UrlTree> {
14          return this.authService.user.pipe(
15              take(1),
16              map(user => {
17                  const isAuth = !!user;
18                  if (isAuth) {
19                      return true;
20                  }
21                  return this.router.createUrlTree(['/auth']);
22              }));
23      }
24  }
```

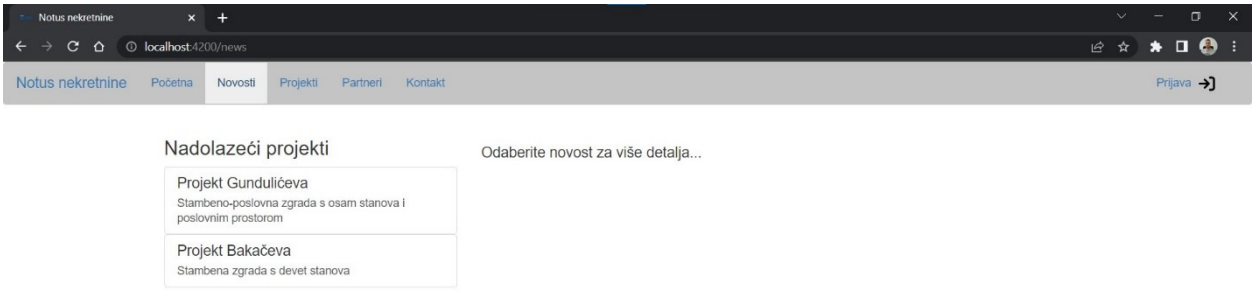
Slika 4.20. Kod *auth.guard.ts* komponente

4.5 Konačan izgled web aplikacije

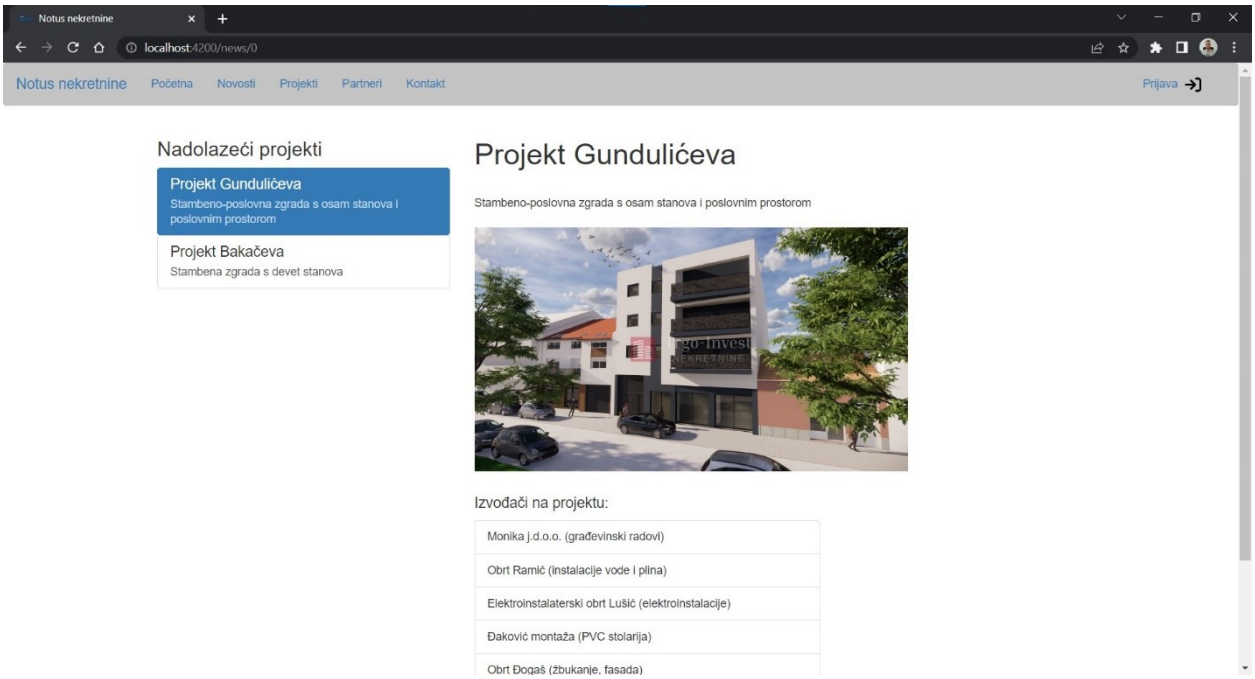
Nakon kreiranja svih spomenutih komponenti, direktiva, servisa i pravilno odrađenog usmjeravanja, web aplikacija poprimila je svoj konačan izgled. U nastavku se može vidjeti izgled svih pojedinačnih linkova unutar aplikacije (početna, novosti, projekti, partneri i kontakt).



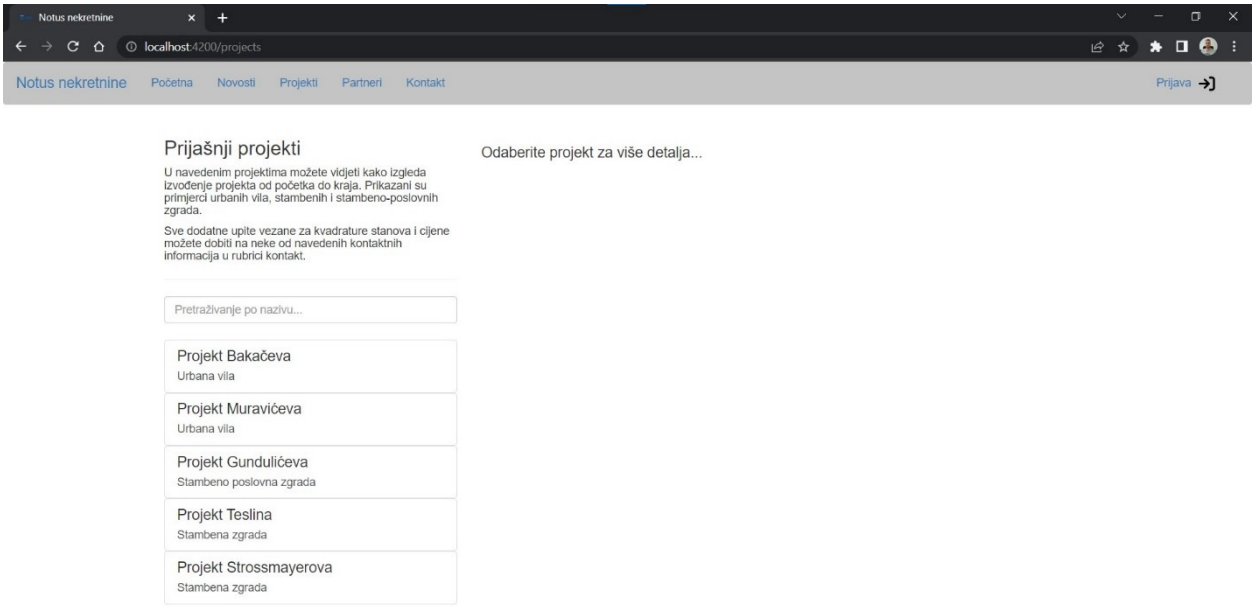
Slika 4.21. Početni izgled web aplikacije (*home* komponenta)



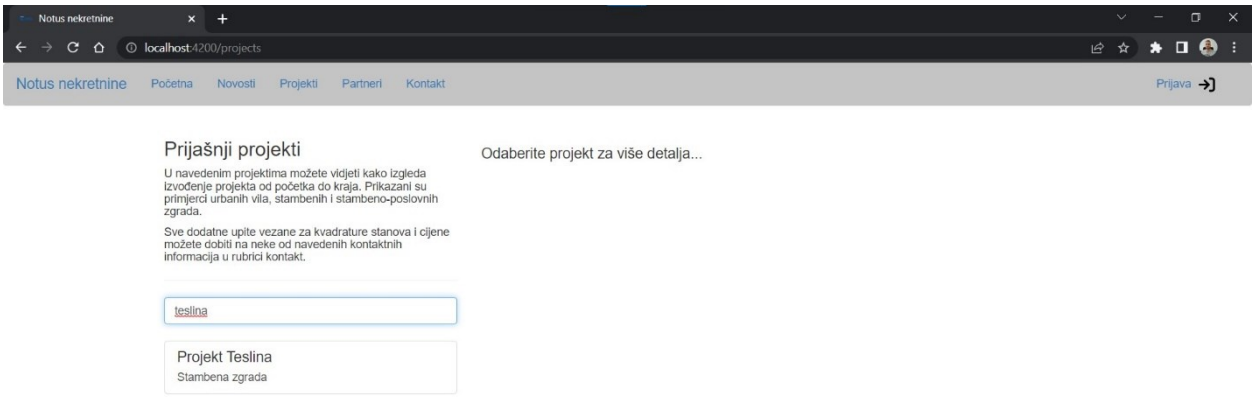
Slika 4.22. Izgled *news* komponente



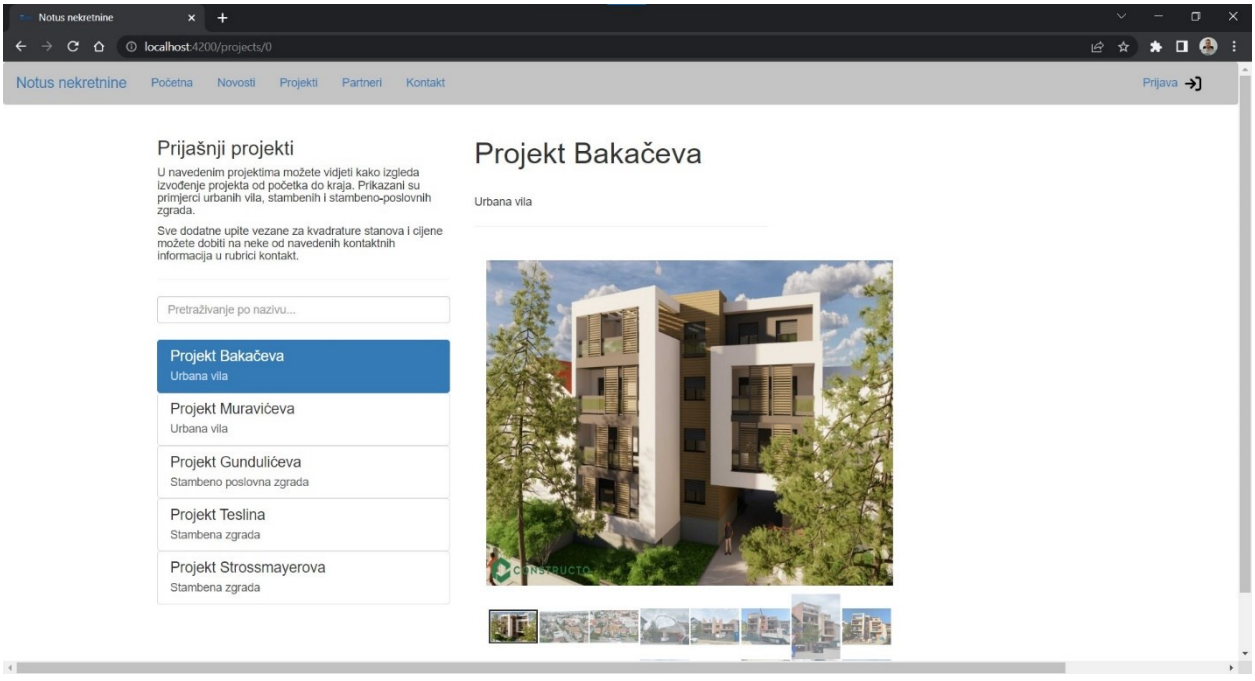
Slika 4.23. Izgled *news* komponente nakon odabrane novosti



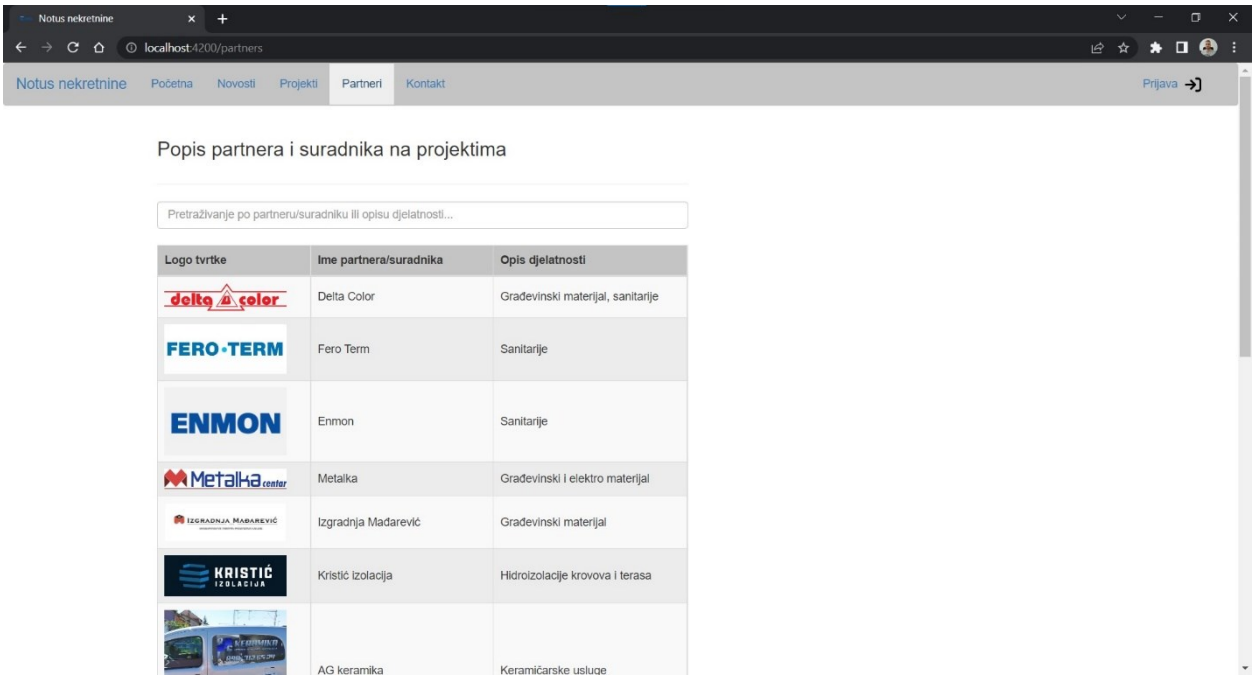
Slika 4.24. Izgled *projects* komponente



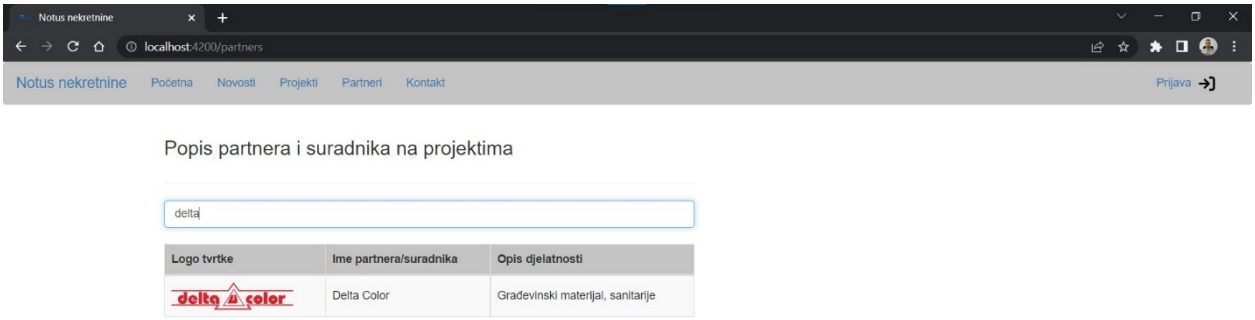
Slika 4.25. Izgled *projects* komponente prilikom pretraživanja projekata



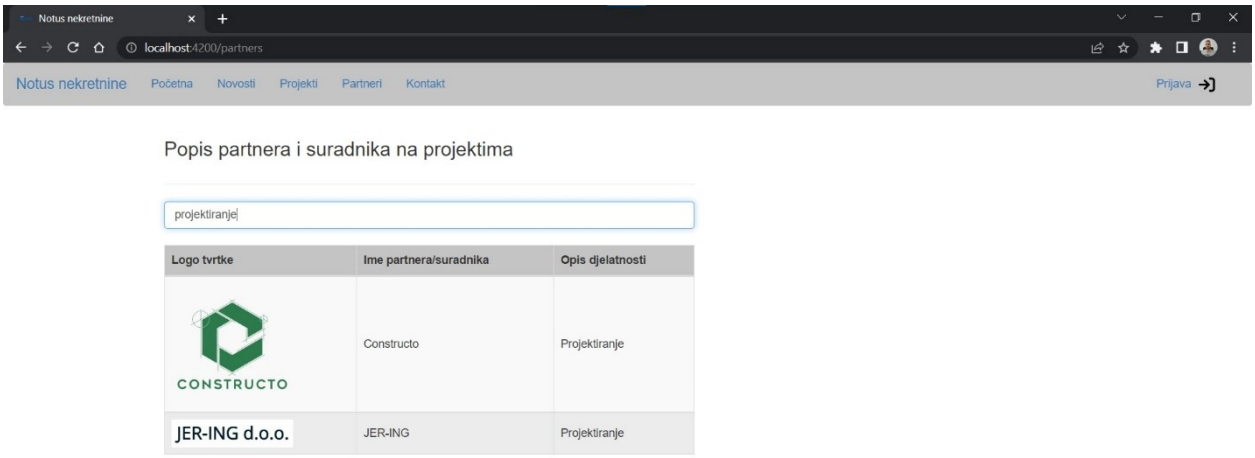
Slika 4.26. Izgled *projects* komponente nakon odabranog projekta



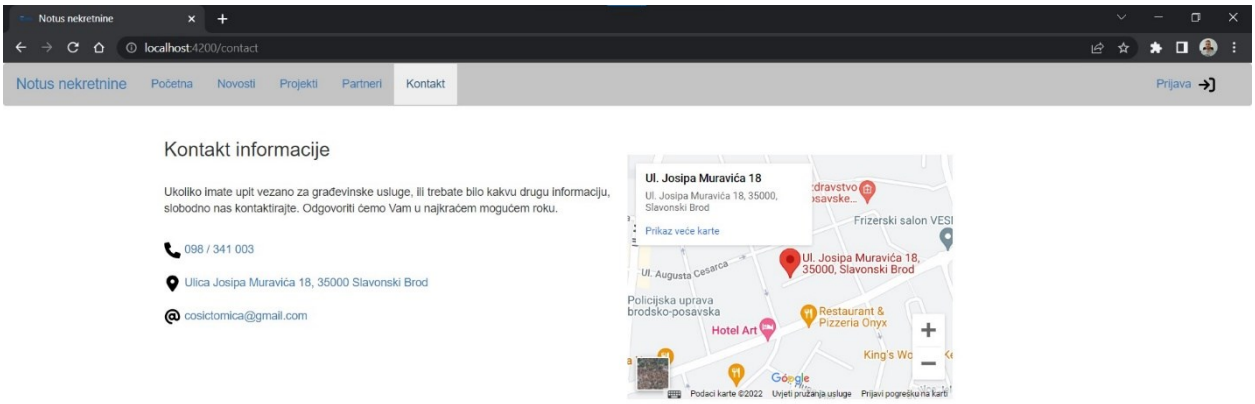
Slika 4.27. Izgled *partners* komponente



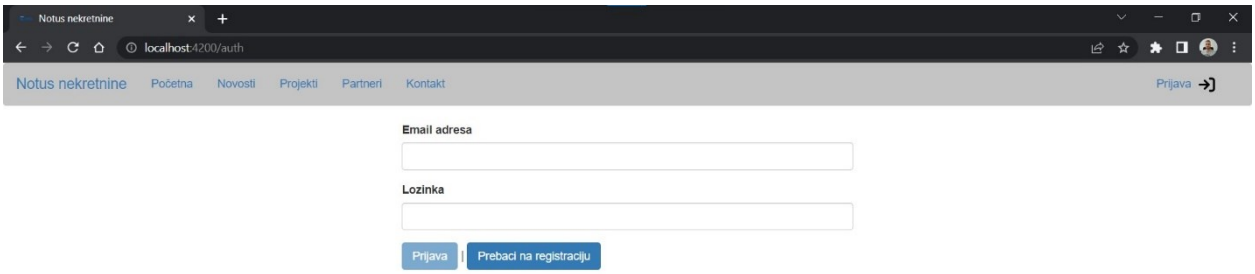
Slika 4.28. Izgled *partners* komponente nakon pretraživanja prema nazivu partnera/suradnika



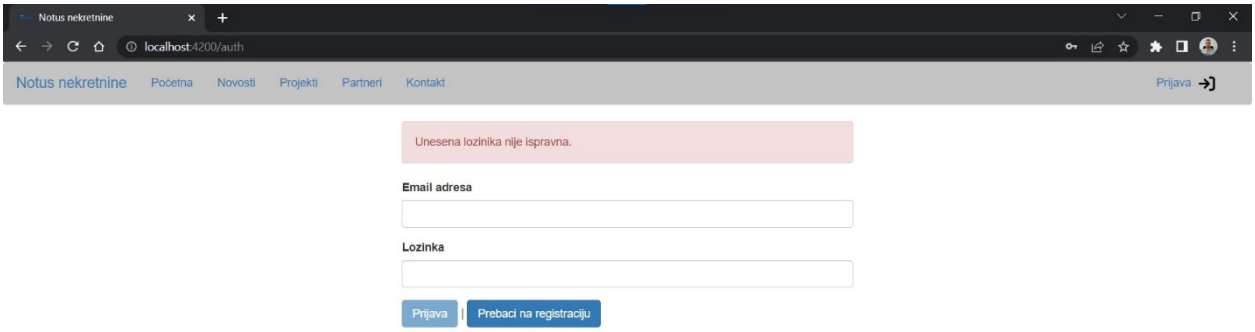
Slika 4.29. Izgled *partners* komponente nakon pretraživanja prema opisu djelatnosti partnera/suradnika



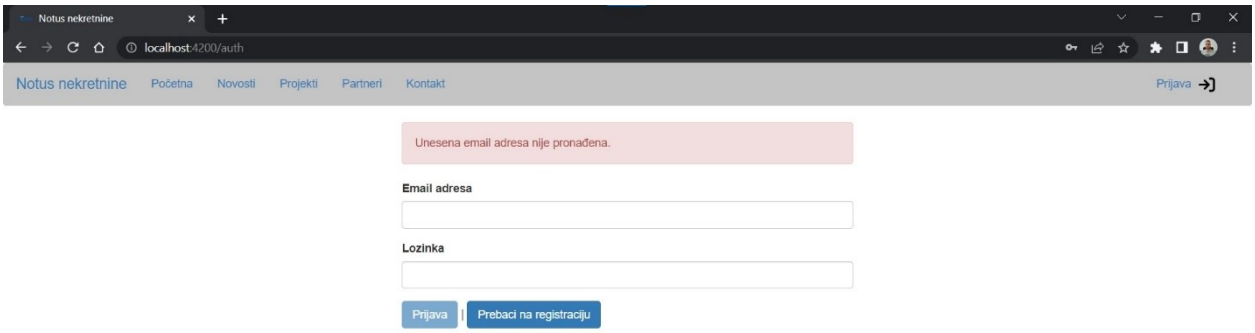
Slika 4.30. Izgled *news* komponente



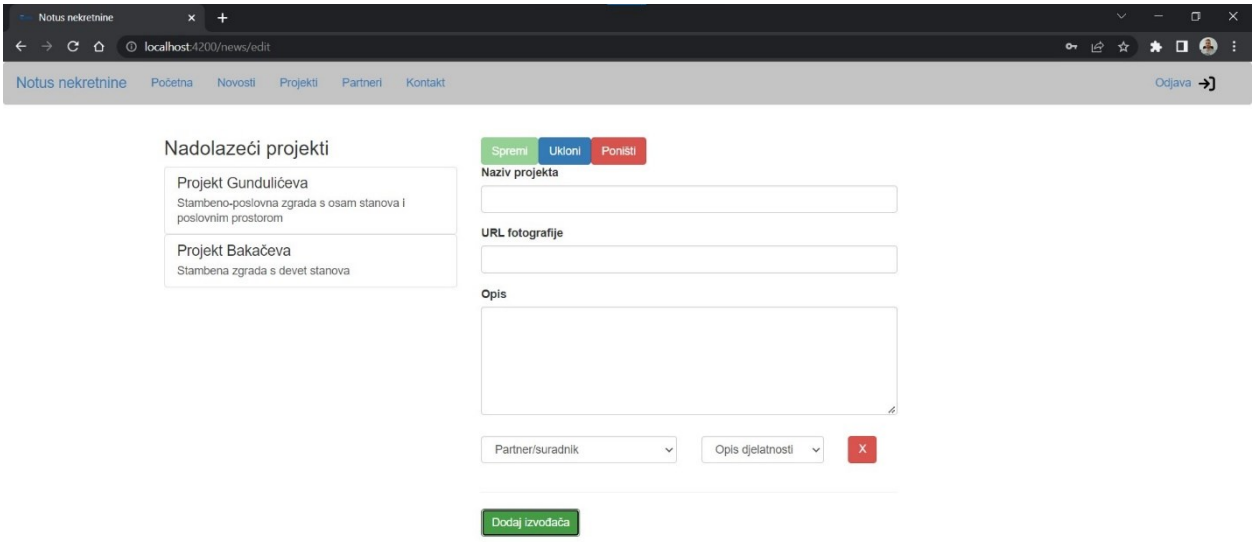
Slika 4.31. Izgled *auth* komponente



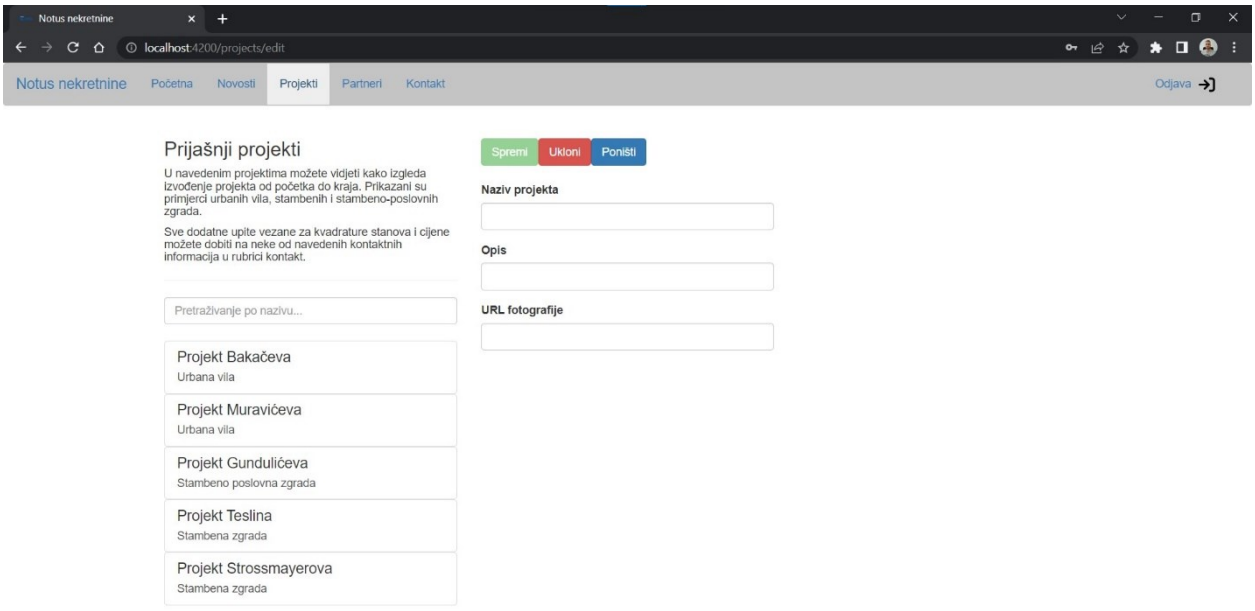
Slika 4.32. Izgled *auth* komponente nakon neispravnog unosa lozinke



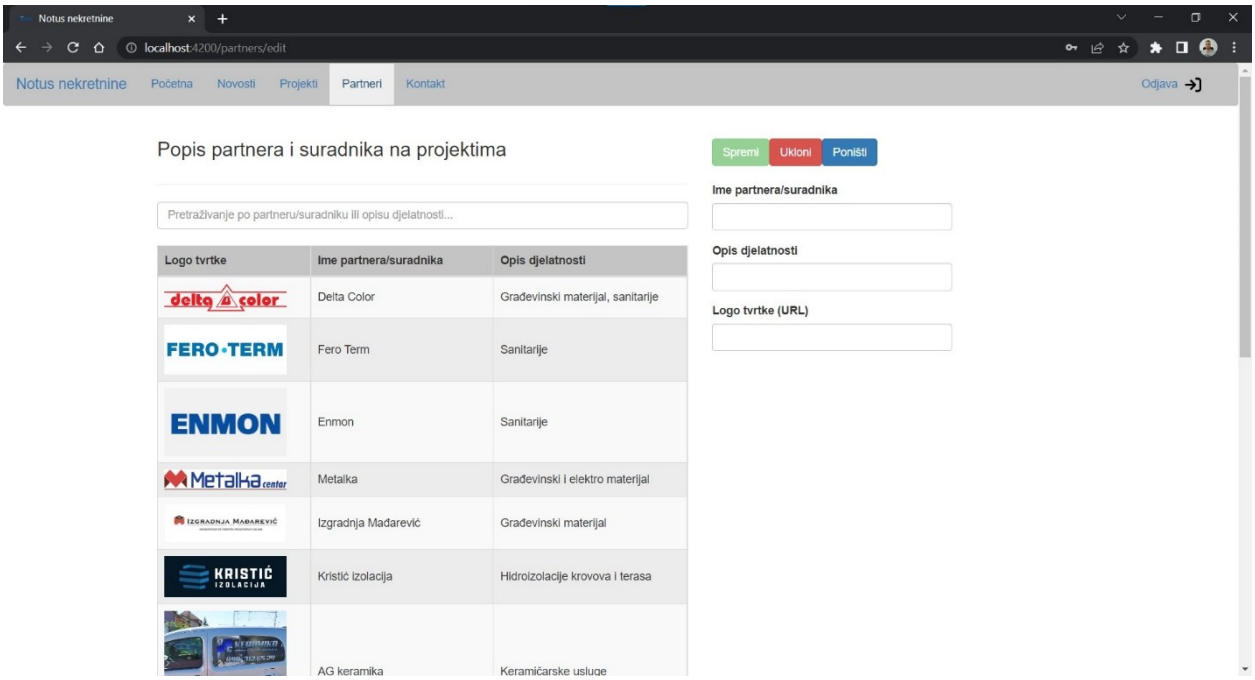
Slika 4.33. Izgled *auth* komponente nakon unosa neregistrirane email adrese



Slika 4.34. Izgled *news/edit* komponente



Slika 4.35. Izgled *projects/edit* komponente



Slika 4.36. Izgled *partners/edit* komponente

5. Zaključak

Tehnološki svijet iz dana u dan se sve više mijenja. Digitalizacija velikim koracima ide prema naprijed i može se uvidjeti u svim gospodarskim granama. Isto tako, uočena je potreba za digitalizacijom jednog dijela poslovanja građevinske tvrtke, što je bilo opisano u ovome završnom radu. Bilo je potrebno ostvariti informativni sadržaj web aplikacije, gdje će se uz osnovne podatke o građevinskoj tvrtki i njenim prethodnim projektima, potencijalni kupci moći informirati i o budućim projektima. Samo rješenje ostvareno je korištenjem HTML, CSS, Bootstrap i TypeScript tehnologija, što je objedinjeno u Angular frameworku. Web aplikacija omogućuje jednostavan pristup svim potrebnim informacijama koje kupci mogu tražiti o jednoj tvrtki.

Literatura

[1] *Constructo*, dostupno na http://constructo.hr/ [stranica posjećena 20.7.2022.]

[2] *DeltaColor,* dostupno na https://www.deltacolor.hr/ [stranica posjećena 20.7.2022.]

[3] *Kristić izolacija*, dostupno na https://www.kristic-izolacija.hr/ [stranica posjećena 20.7.2022.]

[4] *Metalka centar*, dostupno na http://www.metalka.hr/ [stranica posjećena 20.7.2022.]

[5] *Đaković montaža*, dostupno na https://www.djakovic-montaza.hr/o-nama/
[stranica posjećena 20.7.2022.]

[6] *HTML*, dostupno na https://html.com/ [stranica posjećena 15.8.2022.]

[7] *CSS*, dostupno na https://www.w3.org/Style/CSS/Overview.en.html
[stranica posjećena 15.8.2022.]

[8] *Bootstrap*, dostupno na https://getbootstrap.com/ [stranica posjećena 15.8.2022.]

[9] *JavaScript*, dostupno na https://www.javascript.com/ [stranica posjećena 15.8.2022.]

[10] *TypeScript*, dostupno na https://www.typescriptlang.org/ [stranica posjećena 15.8.2022.]

[11] *Node.js*, dostupno na https://nodejs.org/en/about/ [stranica posjećena 18.8.2022.]

[12] *Angular*, dostupno na https://angular.io/ [stranica posjećena 18.8.2022.]

[13] *Firebase*, dostupno na https://firebase.google.com/ [stranica posjećena 4.9.2022.]

[14] *Firebase Realtime Database*, dostupno na https://firebase.google.com/docs/database
[stranica posjećena 4.9.2022.]

[15] *Firebase Authentication*, dostupno na https://firebase.google.com/docs/auth
[stranica posjećena 4.9.2022.]

Sažetak

Cilj ovog završnog rada je kreiranje web aplikacije za upravljanje građevinskom tvrtkom. Web aplikacija osim što nudi informacije o osnovnim podacima tvrtke, o poslovanju tvrtke, odnosno prethodnim i budućim projektima, web aplikacija omogućuje direktoru i upravljanje izvođačima na projektima. Angular framework korišten je za frontend dio. Teorijski opis Angular elemenata koji su korišteni u ovome radu nalazi se u poglavlju *Tehnologije*, a njihova izrada i primjeri iz samoga projekta nalaze se u poglavlju *Izrada aplikacije*. Dizajn web aplikacije ostvaren je korištenjem CSS-a i Bootstrap-a. Na kraju treba dodati da je spremanje novosti, odnosno backend dio projekta, ostvaren pomoću Google Firebase alata *Realtime Database* i da je autorizacija korisnika ostvarena pomoću Google Firebase alata *Authentication*.

Ključne riječi: Angular, Authentication, Bootstrap, CSS, Google Firebase, Realtime Database

Abstract

Angular web application for guiding civil engineering company

Main goal for creating this web application is to help civil engineering company in its business. Web application does not only offer basic information about the company, some of previous and future projects, it also enables CEO of the company to manage performers on the projects. Angular framework was used for the frontend part. Theoretical description of Angular elements which were used in this final paper can be found in chapter *technologies*, and creation of them and examples from the project can be found in chapter *coding the application*. Design of web application was achieved by using CSS and Bootstrap. In the end, we need to mention that saving the news, or the backend part of the project, was accomplished with Google Firebase tool *Realtime Database*, and the user authentication was accomplished with Google Firebase tool *Authentication*.

Keywords: Angular, Authentication, Bootstrap, CSS, Google Firebase, Realtime Database

Životopis

Autor ovog završnog rada je Marko Ćosić, rođen 2. siječnja 2000. godine u Slavonskom Brodu. U istome gradu pohađa prirodoslovno-matematičku gimnaziju Matija Mesić koju završava 2018. godine. Nakon srednje škole, upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

Marko Ćosić

_____